Three Essays in Statistical Learning

Dissertation
submitted to the
Faculty of Business, Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktorin der Wirtschaftswissenschaften, Dr. oec.
(corresponds to Doctor of Philosophy, PhD)

presented by

Vanessa Kummer
from Liechtenstein

approved in October 2020 at the request of

Prof. Dr. Karl Schmedders
Prof. Dr. Thomas Lontzek

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, 21.10.2020

The Chairman of the Doctoral Board: Prof. Dr. Steven Ongena

Acknowledgments

I express my deepest gratitude to my supervisor, Karl Schmedders, for his guidance, patience, and lasting support on this journey. A big thank you also to Thomas Lontzek, for agreeing to be part of my dissertation committee. I am very grateful to Dave Brooks for his invaluable editorial support. I thank my coauthors, my colleagues at the Chair of Quantitative Business Administration, and my friends at UZH and ETH for countless discussions and valuable inputs. Finally, I sincerely thank my family for their encouragement and Michi for always being there for me.

This work is financially supported by the Fonds zur Förderung des akademischen Nachwuchses (FAN) and by the Forschungskredit Candoc.

Contents

General Introduction

Over the last five years, the worldwide number of Google searches for the term "machine learning" (ML) has quintupled; over the last ten years it has increased tenfold (`https://www.google.com/trends`). ML is on everyone's lips nowadays—and this is not surprising as we find it everywhere in our everyday lives: social media services use it to suggest people we may know, it identifies whether an email is spam or not, commercial enterprises use it to recommend products or movies, chatbots on websites serve as customer support representatives, and automated passenger transport is just about to take off. But why has ML become so popular? How is ML different from statistics, which has been used in the social sciences for an age?

Broadly speaking, ML or statistical learning methods combine approaches from statistics and computer sciences, with the main areas of focus being prediction, classification, and clustering tasks. Different from statistics, the main focus in ML is not on *inference* but on finding the best *predictive* function. And because the main goal is accurate prediction, the models are often so complex that they are no longer interpretable. In order to train them while minimizing the risk of overfitting—that is, finding a function that also predicts well with new data—a lot of data (samples) is needed. The data flood of the last few years has been a crucial driver in ML's increasing popularity. In addition, increases in readily available computing power mean that ML can nowadays be used by almost anyone, and thus explain its spread to numerous fields—including, increasingly, to the social sciences.

This dissertation focuses on the use of statistical learning for business and economics. But it also points out pitfalls that accompany the increasing use of ML, and outlines approaches via which one particular pitfall—undesired bias—might be countered.

In Paper 1 we address the fact that much new accommodation in Switzerland has—during the construction boom sustained by the persistence of low interest rates—been built in regions in which a strong increase in demand was expected but has not materialized. This has

resulted in high vacancy rates in these regions. Usually, demand for real estate is estimated by looking at transaction data[1], but the problem with this data is that it captures supply rather than demand. In order to better understand what people are really searching for, we propose analyzing accommodation seekers' search subscriptions to relevant online platforms. The data set exhibits many missing values though. In practice, samples that are not complete are often simply deleted or missing values are imputed with their variables' means. We show that these approaches lead to a huge decrease in the quality of analyses that are based on such data sets. As an alternative, we propose imputation approaches based on unsupervised learning.

Unsupervised learning involves finding clusters of observations that are similar in terms of their features/covariates. In order to minimize possible distortions resulting from training the models on complete observations first, we impute the missing values in an iterative manner: once the first model is trained, we impute the missing data of observations exhibiting only one missing value. We then re-train the model on the complete and imputed data in order to impute the missing data of observations exhibiting two missing values, and so on. In addition, we perform imputation multiple times and each imputed value is averaged (pooled) over the iterations in order to incorporate the uncertainty resulting from imputation.

Our unsupervised learning algorithms for imputation (k-means clustering, Kohonen networks, and matrix factorization) clearly outperform mean imputation. Of the three unsupervised learning models, matrix factorization performs best, in terms of imputation accuracy but also in terms of robustness.

Matrix factorization is also the core of Paper 2. The algorithm became very popular for recommender systems due to the Netflix Prize (Koren et al., 2009). Most firms known to use recommender systems nowadays are Internet firms, including Amazon, Netflix, and LinkedIn. But traditional firms could also enhance their competitiveness by using recom-

[1]See, for example, `https://www.wuestpartner.com/immobilienbewertung/hedonische-bewertung`.

mender systems—by cutting costs related to sales personnel or by increasing their customers' perception of brand value by making them feel understood/known.

Paper 2 therefore presents how the concept of recommender systems can be applied to brick-and-mortar travel agencies. Travel agencies have collected a lot of data about their customers over recent years but the majority still do not know how to make use of it. We therefore provide a manual on how to set up recommender systems as a supportive tool, based on customers' travel histories and user as well as item metadata. The respective performance of the recommender system approaches presented—neighborhood models and matrix factorization—is then compared, first, to that of the popularity model and, second, to a recommender suggesting customers' most recently booked trip.

The often deployed popularity-based approach clearly performs worse than the recommender systems presented. Beating the second baseline, however, proves very difficult. This can, though, be due to the fact that this approach has often been used in the past and therefore that its legacy is reflected in the data. Either way, this approach does not provide significant added value because it will never recommend new regions that might also be of interest to customers but that they are yet to travel to.

This paper presents the case of a specific firm/industry. But when one observes how tech and retail are merging—Amazon and Whole Foods Market being an illustrative example—it is undeniable that firms need to start adopting AI-powered technologies in order to gain a competitive edge.

Using ML techniques also has its pitfalls though. When undesired bias is present in the training data, the resulting model will reflect that bias. This is especially concerning since ML systems are often not deployed in isolation but rather as part of a larger system (for example, facial recognition systems that are part of criminal detection pipelines), therefore making it even more difficult to recognize undesired existent bias. Knowing how such bias can enter ML models and how to get rid of it is, therefore, crucial when working with ML systems.

In Paper 3 we give an overview of how undesired bias enters models trained with ML techniques and of what concepts of fairness exist, how to measure the degree of unfairness, and how to defeat it. As a concrete example we look at automatic speech recognition (ASR) and its bias toward speakers' accents. ASR systems are often criticized in the literature for decoding the speech of certain groups of people more accurately than that of others. We analyze how unequal amounts of speech data per group available for training influences (un)fairness. This issue is especially concerning, as imbalances are difficult to correct due to uncountable numbers of groups and the immense costs of collecting speech data.

Our analyses show that speech recognition models perform well in decoding speech that exhibits accents that are close to those the models were trained with, but less well for accents that are less represented in the training corpus. Since the outcome (decoding accuracy) therefore disproportionally impacts people with certain sensitive attribute values, ASR models trained on an imbalanced speech corpus are said to suffer from disparate impact.

In addition, we analyze how much information about speakers' (potentially sensitive) attributes is contained in the features that are extracted from speech (more concretely, from audio waveforms) and used to train ASR models. We show that models that are trained on these so-called mel-frequency cepstral coefficients (MFCCs) suffer—due to the very nature of how MFCCs are set up—from "disparate treatment": the distance between MFCCs extracted from the speech of speakers that share the same accent is clearly smaller than that for speakers that have different accents to one another. Disparate treatment implies that a person's accent—but given how MFCCs are set up also other attributes we consider to be sensitive such as a person's gender, for example, are used—if indirectly—when training ASR models.

One approach to making ASR models fairer is to include a wrapper in the model that makes the MFCCs of speakers whose accent is underrepresented in the training corpus (in our case, Indian English) resemble that of speakers whose accent was predominant in the speech

used to train the ASR model (here, American English). We therefore adapt the technique of generative adversarial networks (GANs), whose purpose is to generate output that is indistinguishable from its non-generated, real equivalent. GANs were originally designed to generate real-looking images from vectors of random numbers. In our case, we train the generator to transform Indian English speakers' MFCCs into American English ones. The goal of the "discriminator" is to then identify if MFCCs originate from Indian English speakers—and have been modified by the generator in order to *resemble* the MFCCs of American English speakers' speech—or if they originate from American English speakers. In addition, we need to ensure that the content of speech remains the same after the generator has modified it. We therefore need the network to be cycle-consistent, meaning that if we transform something from one accent into the other, and then transform that output back to the original accent again, we should get something that is close to the original input. This can be achieved using CycleGAN.

Once the network is trained, its generators can be used to translate the desired MFCCs. Feeding modified MFCCs of Indian English speakers back into the system for decoding should result in lower word error rates for speech in that accent. So far, this has, however, not been achieved. From what we have experienced and have read about training GANs, we learned that finding an equilibrium between the two competing neural networks is a very difficult task given that their cost functions are non-convex and the parameter space is extremely high-dimensional. But also that we are confident that with further model tuning we will achieve convergence, leading to the desired results.

Overall, this dissertation provides both methodological and practical contributions to the use of ML in economics and business. In 2018, Teich published an article in *Forbes Media*, stating that, "The technologies and techniques of AI and ML are still so new that the main adopters of the techniques are the large software companies able to hire and to invest in the necessary expertise" (Teich, 2018). This dissertation proves that brick-and-mortar

companies too would benefit from applying ML techniques, and provides a comprehensive guide to how. Besides improving on classical approaches to analyzing data and therefore getting more information out of that data, ML methods can also be used to extend data bases for such analyses by predicting the missing values of samples that would otherwise just be disregarded. Despite these and many other benefits, however, ML is also associated with certain risks. In this dissertation we discuss the issue of algorithmic bias and show how to approach this problem. Thus, even though we look specifically at the example of speech recognition, the findings can be extended and applied to many other problems.

Three Essays in Statistical Learning

Housing Market Analyses—Learnings from Search Subscriptions[1.1]

Vanessa Kummer, University of Zurich
Andy Egger, Realmatch360
Maik Meusel, University of Zurich
Karl Schmedders, University of Zurich and IMD

Abstract

The dramatic increase in the amount of vacant accommodation in some regions of Switzerland and the simultaneous housing shortage in others are the result of not knowing where people want to live and, therefore, of having built accommodation in the wrong locations. In order to better understand what people are searching for, this paper proposes analyzing accommodation seekers' search subscriptions to relevant online platforms. Using search subscriptions allows us to get a better understanding of people's preferences for housing and even to identify unmet demand. Search subscription data, though, usually exhibits many missing entries. Thus, the present paper proposes powerful approaches based on unsupervised learning to impute the data and, therefore, maximize its benefits for housing market analyses.

[1.1]We thank Realmatch360 for providing us with data, and Dieter Marmet from Realmatch360 for his thoughtful support. We are also grateful to Rob Earle, Felix Kübler, Harry Paarsch, Michael Wolf, and seminar participants at the University of Zurich for their feedback on our work.
Vanessa Kummer would like to express her gratitude to the Fonds zur Förderung des akademischen Nachwuchses (Research Talent Development Fund), established by the Zurich University Association, for its financial support.

1.1 Introduction

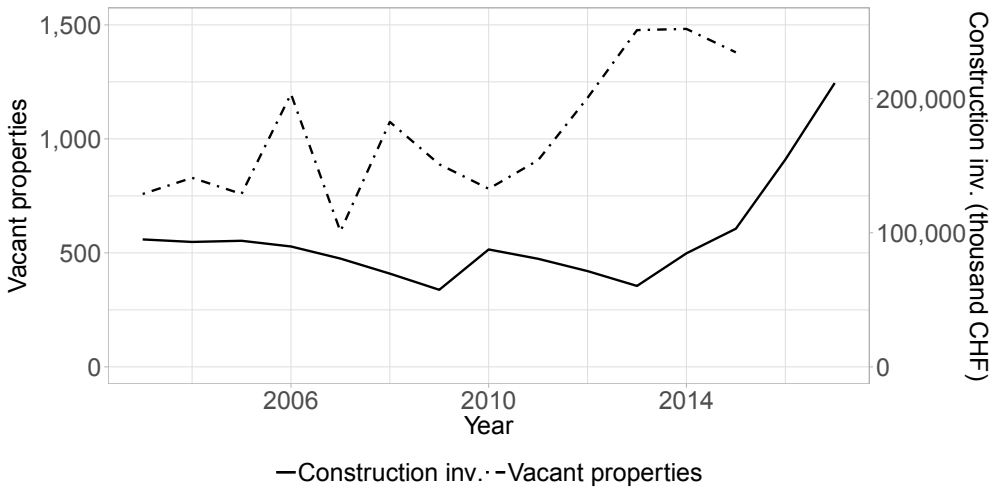
It is 2017, and Swiss newspaper articles on high accommodation vacancy rates and consequent frustration among housing stock owners have been piling up. The persistence of low interest rates has led to a construction boom. The result: in some parts of Switzerland more than 10 percent of the apartments available for rent are currently vacant. Many of these apartments have been built in regions in which a strong increase in demand was expected; but so far, this increase has not materialized. Moreover, many places are vacant because their rents exceed potential tenants' maximum willingness to pay. This is especially true of apartments situated outside of city centers. Both problems result from not knowing what properties people are really interested in.

A prominent example is the region of Olten. Situated at the intersection of the main rail and road links between the major agglomerations of Zurich, Basel, and Bern, the region seems to have substantial potential. And a large amount of accommodation has therefore been built there (see Figure 1.1). But despite expectations of an increase in the residential population, much accommodation in the region remains vacant (Vontobel, 2017; Staehelin and Heiniger, 2017). Due to the increasing supply and the lack of demand for accommodation in regions such as Olten, rents are likely to come under considerable pressure. In the worst case, this could even lead to a housing crisis.

Staub and Rütter (2014) show in their study that housing market output constitutes around 18 percent of overall economic output in Switzerland. But it is not in Switzerland alone that the housing market is of fundamental importance; the United States' housing crisis, resulting from a huge boom and subsequent collapse in housing prices, affected the *global* economy.

Hence, knowing where people actually want to live—and where not—is of great interest. Imagine, for example, an investor or a project developer: construction in regions where there turns out to be a lower demand than expected leads to lower revenues, loss of money, or—in the worst case—to bankruptcy. And it is also important for policy makers to have information on where people actually want to live. In the region of Olten, for example,

Figure 1.1: Construction investments/number of vacant accommodation properties in Olten



The plot shows how—as a result of persistently low interest rates in Switzerland—construction investments in the region of Olten have increased over recent years. Demand for housing in the region has, however, been overestimated, which—due to the increasing supply of housing—has led to an increasing number of vacant accommodation properties. Data source: Swiss Federal Statistical Office (https://www.bfs.admin.ch/bfs/en/home/statistics.html).

further construction zones should, perhaps, not be approved, since an increase in supply will lead to even lower prices for the properties there.

If further construction in inappropriate regions is to be avoided, knowing about demand for real estate and its development is indispensable. The research carried out in the framework of the present paper can help to improve knowledge in this regard.

Usually, demand for real estate is estimated by looking at transaction data. The problem with transaction data, though, is that it captures supply rather than demand. Hence, demand with no corresponding supply cannot be recognized by looking at transaction data. In addition, transaction data provides only a retrospective picture. Instead, this paper proposes that accommodation seekers' subscriptions to relevant online platforms be examined. Data on search subscriptions reveals current information regarding what people actually want in terms of real estate.

Search subscriptions are usually set up when the type of property desired is currently unavailable or when it is known that, because of tremendous demand, the search might take

a while and one does not want to miss any new object that is published on the platform in question (Wüest & Partner, 2015). Both these motivations imply a demand for housing (discussed in Section 1.6) that cannot be covered by the actual supply. Hence, the majority of search subscriptions indicate unmet demand.

The problem with search subscription data is that it suffers from a lot of missing information—for example, some users do not specify how many rooms they would like or what price they would be willing to pay. Often, observations with missing values are deleted. The proportion of complete data is, however, usually rather small, which leads to most available information being neglected. In addition, we do not know if observations that are complete reflect the whole population. Therefore, it is essential to impute the data sets at hand. And it is important to impute them as accurately as possible. Only then can housing market analyses based on search subscription data provide an accurate picture of people's preferences and of the market situation.

The approach proposed in this paper is to search for patterns in the observations that exhibit no missing values, and to apply those patterns to the observations that need to be imputed. To search for these patterns we make use of unsupervised learning. Unsupervised learning involves finding clusters of observations that are similar in terms of their covariates.[1.2] There exist a variety of techniques for unsupervised learning. After some analyses, we decided to apply one very common and rather easy algorithm and two more sophisticated ones—namely, k-means clustering, a neural network, and factorization (the last of which became very popular for recommender systems due to the Netflix Prize (Koren et al., 2009)). The data is imputed in an iterative manner to minimize possible distortions resulting from training the models on complete observations first. In addition, by imputing the missing values multiple times using different models, we incorporate the uncertainty from imputation into the imputed data set.

[1.2]The term "unsupervised" refers to the fact that there are no "labels" on any of the observations in the input data set; only after examining the items in each group might an observer be able to determine tags for the groups that the algorithm found.

The rest of the paper is structured as follows: first, the problem of missing data is discussed, followed by an outline of how we handle it and what measures are used to assess imputation performance. Second, the aforementioned unsupervised learning models are introduced. Before turning to the detailed implementation of the imputation procedure, the data set we use for our analyses is described, and its missing data patterns analyzed. The results are presented in Section 1.5 and discussed in Section 1.6. The paper closes with a brief conclusion.

1.2 Missing Data

The scientific debate about missing data in statistical analyses emerged in the context of surveys. Surveying individuals usually results in sample data sets that suffer from non-response. Most analytical tools cannot handle missing values, which is why they are often deleted by default. But omitting observations with missing data leads to less efficient analyses due to a decrease in sample size. In addition, it might lead to biased results if the respondents and non-respondents systematically differ from each other (Rubin, 1987). Hence, missing values should be *imputed* to mitigate both these issues.

In order for imputation to not "disimprove" the results of subsequent analyses, the mechanisms that cause the data to be missing in the first place have to be *ignorable* (Schafer, 1997). A necessary condition for "ignorability" is that the missing data is either *missing at random* or *missing completely at random* (Little and Rubin, 2014).

1.2.1 Missing Data Mechanism

Missing data is said to be *missing completely at random*, MCAR, if the fact that it is missing depends neither on its own value nor on the non-missing values of the respective observation. If the data is MCAR, respondents and non-respondents do not systematically differ from each other. Deleting observations with missing values does not lead to biased estimates in this

case, but the imputation of these missing values might improve later analyses since that increases the sample size (Little and Rubin, 2014).

If the "missingness" of a value depends solely on the complete variables of the same observation, the value is *missing at random*, or MAR. If the fact that an item is missing depends on the value it would take if it were not missing, the item is said to be *not missing at random*, or NMAR. In such a case, the missing data mechanism is not ignorable and has to be explicitly included in the imputation procedure (Little and Rubin, 2014).

To make sure missing data can be imputed without considering the missing data mechanism, one would like to test whether the data is MAR or MCAR, as opposed to NMAR. There is, however, no formal way of testing whether data is missing at random or not. Nonetheless, a test—introduced by Little (1988)—is available to assess whether missing data is MCAR as opposed to MAR or NMAR. The author proposes a single global test statistic that uses all of the available data with a null hypothesis stating that the data is MCAR. If the null hypothesis holds, one can ignore the missing data mechanism. If the null hypothesis has to be rejected, one has to rely on theoretical reasoning to motivate the MAR assumption (Little, 1988).

1.2.2 Multiple Imputation

Rubin (2004) notes that, in general, imputation of a *single* value for a missing one does not necessarily lead to that value being correct. The problem is that single imputation does not take into account the uncertainty inherent in imputation. After being imputed, data is treated as if it were the true value. The author therefore suggests an approach called *multiple imputation*: each missing value is imputed m times, resulting in m complete data sets. By imputing multiple times, one can account for the aforementioned uncertainty and for the range of values that the true value could have taken (Rubin, 2004).

After imputing a data set m times, the resulting measures of interest are combined according to what are known as *Rubin's rules*. Usually, the mean over all m estimates is taken as a

single value for the measure of interest (Rubin, 1987).

According to Rubin (1978), multiple imputation provides valid results as long as the mechanism that causes the data to be missing is ignorable. A low number for m, usually five, is then considered sufficient to get some idea about the variability in the data set caused by missing values (Rubin, 2004). Taking into consideration this potential variability caused both by missing data and by the respective imputation method is a major advantage of multiple imputation over single imputation (Rubin, 1987).

1.2.3 Notation

In the following, we will refer to subsets of our data D by using the notation $D_t^{m,s}$, where

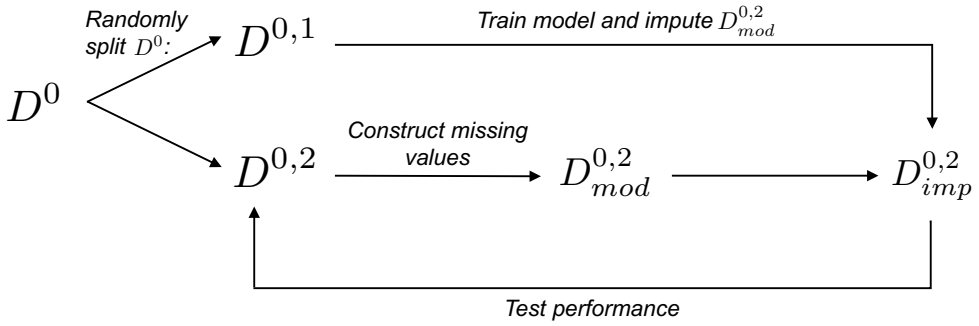
- D^m denotes the set of records that have $m = 0, 1, 2, \dots$ missing values,
- D_t^m refers to the set D^m having been imputed ($t = imp$ in the case of $m = 1, 2, \dots$), or modified ($t = mod$ in the case of $m = 0$) by the dropping of values,
- $D^{m,s}$ denotes possible subsets $s = 1, 2, \dots$ of D^m.

1.2.4 Iterative Imputation and Imputation Performance

The process of performance testing in the training step is illustrated in Figure 1.2. The idea is to compare the models by looking at records that have no missing values. This allows us to measure the imputation accuracy of the algorithms used.

In a first step, we isolate 10 percent (randomly) of the observations in D for final performance testing. These observations are not used for model training (and thus constitute the "hold-out" data (set)). From this test data, we randomly remove 5 percent of the data and store it for later comparison to the imputed values.

From the 90 percent of data not used for final testing, we pick D^0, the records that have no missing values. Then, we randomly split D^0 into two subsets—a training set, $D^{0,1}$, and a test set, $D^{0,2}$. In $D^{0,2}$, we delete values such that the sparsity pattern equals that of D (see Figure 1.3). The modified data set is now called $D_{mod}^{0,2}$.



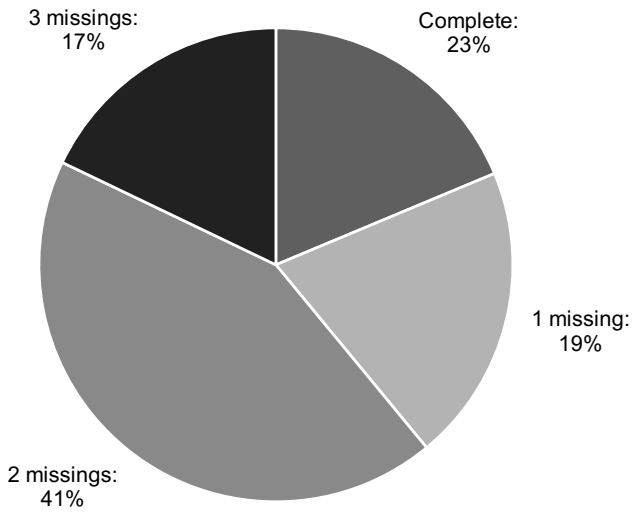
The figure illustrates how we tune the imputation models. We only consider data with no missing values, D^0, for tuning. The data set is randomly split into two subsets. From one of the subsets we randomly remove values and store them for later comparison. With the other subset we train the model. The subset with the removed values is then imputed using the trained model, and the imputed values are compared to the previously removed and stored ones. This process is repeated several times for each model parametrization to reduce the risk of having distorted subsets.

Next, we train our models using $D^{0,1}$, and impute all records of $D^{0,2}_{mod}$ that have one missing value. The three methods used will be explained in Section 1.3.

Since, ultimately, we impute the missing values using models trained on the complete records, D^0, and since we do not know if D^0 is a representative sample of the whole population, we impute the data set iteratively. This means that we train the models not only using D^0, but also on the available imputed data (see Figure 1.4). The motivation to do so is the following: by training the models iteratively on the imputed data and, thereby, including as much information as possible in the models, possible distortion impacts are minimized.

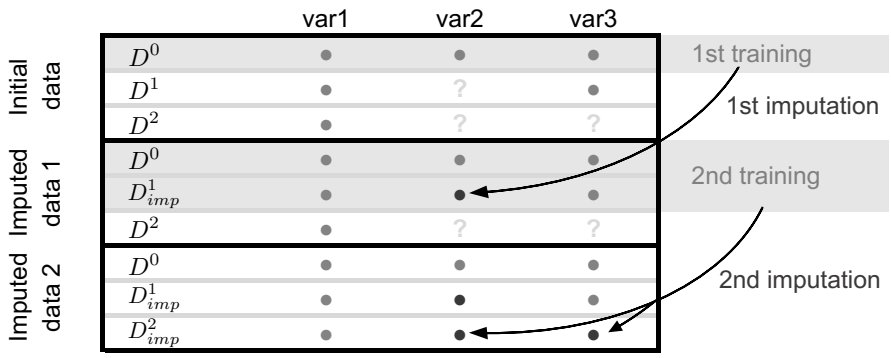
Our analyses using the complete records—D^0—only, show that the performance of iterative imputation is not significantly different from that of non-iterative imputation. However, in these pre-analyses, the first training set is of course a representative subset of D^0 since it is a random subset of it. But we do not know if D^0 is a random subset of D. One can think of an example where all complete observations stem from people from a wealthy region. If the model would be trained on these observations alone, people from other regions' imputed willingness to pay would tend to be too high.

Figure 1.3: Missingness distribution of our data set



The plot shows that the majority (41 percent) of observations in our data set have two missing values, D^2. Another 23 percent of the observations have no missing values, D^0. These observations are used for tuning the models and, later, build the base for the model that aims to impute D. An additional 19 percent of the observations in our data set have one missing value, D^1, whereas only 17 percent have three missing values.

Figure 1.4: Iterative imputation process



The illustration shows how the iterative imputation process works: The first training is based on complete data, D^0 only. With the trained model, we impute D^1, resulting in the imputed D^1_{imp}. The next training is based on D^0 and D^1_{imp}, resulting in a model which we use to impute D^2. This process is repeated until all observations are imputed. The motivation for this iterative imputation process is to get rid of possible distortions from D^0.

Hence, the next step is to train the model on the initial training set and the records that have been imputed, and impute records that have two missing values per row. This procedure is repeated until all missing values are imputed.

The imputed data set is then compared to $D^{0,2}$ in terms of the normalized mean squared error (NMSE)

$$\text{NMSE} = \frac{1}{\mu} \frac{(D_{imp}^{0,2} - D^{0,2})^2}{\lambda}, \tag{1.1}$$

where μ is the mean of the respective variable, and λ equals the number of values that have been imputed in that column.

The imputation procedure is repeated 10 times for each of the imputation approaches and for all reasonable parameter constellations. In every iteration, the data is split randomly into a training set and a test set. The resulting normalized mean squared errors are then averaged over the training–testing random splits. This approach of testing is referred to as repeated random subsampling (Dubitzky et al., 2007). The advantage of this method is that, unlike for example in k-fold cross-validation, the test sets overlap. According to Molinaro et al. (2005), this can substantially reduce the variance of the split-sample error estimation. The authors also report that 10 random training–testing splits are sufficient to realize most of the achievable reduction in variance. The disadvantage of this method is that it may be the case that an observation never or always appears in the training set or test set. But with a large enough number of repetitions, such extreme cases would be unusual (Molinaro et al., 2005).

The procedure above is applied to find the optimal parameters for each model and, following, to train the models. The final performance is then tested on the 10 percent hold-out data set. We therefore impute the missing values of the hold-out set using the previously trained models, and compare the imputed values to the ones previously removed. To measure the performance, we again apply the NMSE from Equation (1.1).

1.2.5 Mean Imputation as Benchmark

A widespread imputation technique involves replacing missing values with the respective column mean. *Mean imputation* has the benefit of not changing the sample mean. Obviously however, its drawback is that it ignores any correlations between the variables that are imputed and the measured variables. Mean imputation may therefore be an attractive technique for univariate analyses, but it becomes problematic for multivariate ones. Nevertheless, we use mean imputation as our benchmark since it is a widespread imputation technique and can serve as a lower bound for the imputation performance. In the following section, we present the imputation techniques—which are based on unsupervised learning—that we use in our analyses.

1.3 Methods

In order to find patterns in the data and to be able to describe associations, we apply unsupervised learning. One of the most common unsupervised learning methods is *k-means*, a clustering algorithm proposed by Lloyd (1982).[1.3] K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster that is nearest to that observation (Lloyd, 1982).[1.4] Clustering D^0 using k-means hence results in k clusters, each having a so-called cluster centroid, which is the mean of the data in that cluster. In order to impute the records that have missing values, we substitute the missing values by the nearest (in terms of Manhattan distance) cluster's centroid values. As explained earlier, the clustering, and following the imputation, is carried out iteratively.

The only parameter that needs to be tuned in k-means clustering is the number of clusters, k. To optimally balance out bias and variance, we choose k such that the NMSE is minimized.

[1.3] The algorithm had already been introduced by Stuart Lloyd in 1957, but was not published outside of Bell Labs until 1982.

[1.4] The pseudocode of k-means and the following algorithms can be found in the appendix to the present paper.

We also analyze the performance of *Kohonen networks* as an example of the class of artificial neural networks (ANNs) that are often used. The algorithm introduced by Kohonen (1982) aims to produce a low-dimensional representation of the input data. Like most ANNs, Kohonen networks operate in two modes: *training* and *mapping*. Training builds the map using the input data, while mapping classifies new input vectors. The map space, consisting of neurons, is defined beforehand. In our case, the nodes can be arranged in either a hexagonal or a rectangular grid. Each neuron is associated with a weight vector, which indicates the position in the input space. In the training step, weight vectors are moved toward the input data (reducing a distance metric[1.5]) without spoiling the topology induced from the map space. Thus, the Kohonen network produces a low-dimensional representation of the input data (Kohonen, 1982). Once the network is trained using D^0, the missing values can be predicted by ascertaining which neuron's weight vector of the trained network is closest (smallest distance metric) to the input vector of interest.[1.6]

The exogenous parameters in Kohonen networks are the number of neurons in the grid, m, as well as the topology of the grid. Again, the optimal parameter combination is the one that minimizes the NMSE.

Another unsupervised learning approach is *matrix factorization* (MF). MF has a long history, but only became better known after Lee and Seung (1999, 2001) published some simple and useful algorithms for factorization. Later, thanks to the Netflix Prize, MF became very popular for deriving recommender systems (Koren et al., 2009). MF involves factoring the multivariate data matrix $X \in \mathbb{R}^{n \times p}$ into two smaller matrices, $W \in \mathbb{R}^{n \times k}$ and $H \in \mathbb{R}^{k \times p}$ (as in Equation (1.2)), in such a way that the product of W and H minimizes the squared error to the non-blank entries of X. Once W and H are trained, their product can be used to predict the missing values in X (Lee and Seung, 1999, 2001).[1.7] The only parameter that

[1.5] The default distance metric in the used R package used—"kohonen" by Wehrens and Buydens (2007)—is "sum of squares".

[1.6] More details on Kohonen networks are available in Appendix B of the present paper.

[1.7] More details on MF can be found in Appendix C of the present paper.

we tune in MF is the decomposition rank, k.[1.8]

$$\underbrace{\begin{bmatrix} 4 & 5 & 1 \\ 2 & & 3 \\ & 3 & 2 \end{bmatrix}}_{X} \approx \underbrace{\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}}_{W} \times \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{bmatrix}}_{H}. \tag{1.2}$$

1.4 Search Subscription Data

Considering real estate portal data for housing market analyses is a relatively new approach and is based on changing search behavior with regard to the search for living space. In the 2016 edition of their yearly survey, *Neue Zürcher Zeitung*/Wüest & Partner (2016) asked 497 people from all over Switzerland—people looking to change accommodation, aged between 15 and 79, and who have Internet access at least once a week—about the importance of sources of information when searching for new accommodation to buy or rent. By far the majority—93 percent of the respondents—stated that they browse online advertisements. A total of 29 percent set up search subscriptions to Internet portals (*Neue Zürcher Zeitung*/Wüest & Partner, 2016).

At first glance, this 93 percent figure appears attractive, but research carried out by the Swiss start-up Realmatch360 has shown that there are several problems with search queries. First of all, since the barrier of searching and making modifications to a search is relatively low, people tend to enter multiple searches, which in turn do not reflect their actual preferences. They use search queries to get an overview of the market rather than explicitly to search for their new homes. The second problem is that personal budgets are often not taken into account when individuals use search queries to merely obtain an overview of housing supply. Often, also, curiosity is the only driver for a search (Martel, 2018).

But what about the 29 percent who set up search subscriptions? Horber (2015) analyzes

[1.8] We analyzed a variety of methods. The three methods explained here turned out best in terms of imputation accuracy as well as in terms of computation time. Results are available from the authors on request.

search subscriptions to real estate portals. In a survey, he asked 1,454 people who had set up search subscriptions about their maximum willingness to pay for real estate objects, as well as about their purchase intentions. He found that only about 10 percent of the subscribers had no real purchase intentions. Also, his analyses show that the actual maximum willingness to pay is only 6.7 percent (median) lower than the maximum price stated in the search subscriptions, and that the majority of subscribers would be willing to pay the price stated for the perfect object (Horber, 2015). People, therefore, take into account their budget when setting up search subscriptions. Hence, search subscriptions provide very accurate information about individuals' demand. Furthermore, since search subscriptions can be created even though no corresponding offer might exist, they are not affected by supply.

1.4.1 Data Characteristics

Realmatch360 provided us with already cleaned search subscription data from the four major Swiss online platforms for housing. The four platforms cover practically all search subscriptions in Switzerland. This fact was established by Realmatch360's analyses, which included search subscriptions from other real estate portals.

The data sets consist of unique search subscriptions from all over Switzerland that were active on any of the four portals on the retrieval dates in question. They include information about location, desired number of rooms, and desired price (see Table 1.1).

The problem with these data sets is that many values are missing. Table 1.1 lists all variables, together with the values they can take and the percentage of missingness of a representative data set (active search subscriptions at 05.11.2016; number of observations: 470,038). In the reference data set, only 17 percent of the records have no missing values. The most frequent pattern is `roomMin` and `priceMax` being observed, and `roomMax` and `priceMin` being missing (27%). In 63 percent of the records both `roomMin` and `priceMax` are observed together.

As we can see from Table 1.1, the two variables with the greatest lack of completeness are `sizeMin` and `sizeMax`. In our analyses (which will be explained later in this paper), we found

Table 1.1: Data set characteristics

Variable	**Values**	**Percentage missing**
`email`	hashed email address	0
`objectType`	{apartment, house}	0
`searchType`	{purchase, rent}	0
`roomMin`	$\{1, \ldots, 8\}$	23
`roomMax`	$\{1, \ldots, 8\}$	61
`priceMin`	[390, 4,500,000]	69
`priceMax`	[400, 4,979,000]	21
`sizeMin`	[20, 515]	74
`sizeMax`	[25, 565]	93
`specificSearch`	{0, 1}	0
`city`	all 2,420 Swiss cities	0
`cityKey`	city keys	0

The table lists all the variables of our data set, as well as the values each variable can take and the percentage of missingness of a representative data set (active search subscriptions at 05.11.2016; number of observations: 470,038). Search subscriptions can be set up either for apartments or for houses. In addition, housing seekers can express their desire either to purchase or to rent. The only variables exhibiting missing values are the ones related to the number of rooms, the size of the accommodation, as well as the price. The two variables with the greatest lack of completeness are `sizeMin` and `sizeMax`. Further variables include the region in which housing is searched for, whether the search is explicit (by entering one zip code only), as well as the email address of the seeker. Data source (for this and all subsequent figures and tables): Realmatch360 data.

that dropping these two variables increases the performance of imputation significantly. In addition, these two variables are highly correlated with the room variables (on average 80 percent), and therefore dropping `sizeMin` and `sizeMax` does not lead to a great loss of information. Also, in later analyses we found that whenever either `roomMin` and `roomMax` or `priceMin` and `priceMax`, respectively, are missing from an observation, performance is very poor, which is why we drop records with this pattern whenever it occurs. Furthermore, because the numbering of cities does not reflect the property's location, we replaced `city`, or rather `cityKey`, by cities' longitudes and latitudes.

Last, the ranges that the variables can take differ quiet a lot from one another (see Table 1.1). Therefore, data is scaled according to the following two steps:

1. All values are centered by subtracting the column means of the values' corresponding columns.
2. After being centered, all values are divided by the standard deviation of the values' corresponding columns.

Scaling is especially important because some of the imputation methods cluster data based on data points' distance from one another. Variables with different scales would be assigned different importance.

1.4.2 Missing Data Analysis

As discussed in Section 1.2.1, for imputation to be valid, the missing data has to be MAR or MCAR. Conducting Little's test, the hypothesis that the data is missing completely at random has to be rejected ($p = 0$). Hence, logical reasoning has to be applied to motivate MAR.

Based on the knowledge that we have gained about real estate platform data in the course of this project, we assume that the completeness of a given record is a measure of the *specificity* of a user's preferences alone—independent of that user's actual preferences. For instance, the propensity of `roomMax` to be missing has nothing to do with its actual value. The magnitude of values' missingness might instead reflect a person's market knowledge or the degree of specificity of the search. Also, people who create search subscriptions have no reason to report their desires inaccurately, so we exclude strategic behavior. Taken together, the data at hand is assumed to be MAR.

1.5 Results

In our representative data set—after deleting `sizeMin` and `sizeMax`—17 percent (that is, 81,117) of the records are complete. Because there are very large price differences between the search subscriptions of people intending to buy and people intending to rent an object, and because there are more search subscriptions made by people who wish to rent, we decided to only consider rental search records for the purposes of this paper. For the same reason, we took apartments as our priority accommodation type. After removing 10 percent of the observations for final performance testing, D^0 is reduced to 68,506 observations for our reference date.

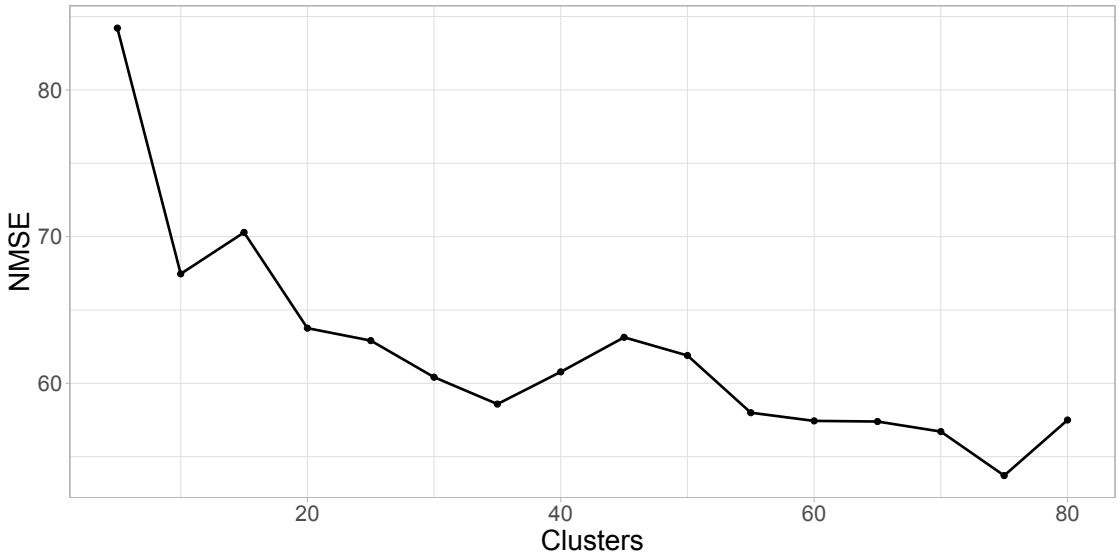
As previously mentioned, in k-means clustering the only exogenous parameter is the number of clusters, k. We analyze k in the range of 5 to 80 clusters and repeat each analysis 10 times—to account for fluctuating performance due to different starting points. According to the resulting NMSEs (see Figure 1.5), performance is best at $k = 75$, which is why we set k to 75 for later analyses. The model turns out not to be really robust though, as its performance heavily depends on the initial random clusters chosen.

To find the optimal parameter combination for the Kohonen network, we perform a grid search cross-validation with the number of nodes m in the range of 1 to 50, and topologies being either *rectangular* or *hexagonal*. The resulting average NMSEs of 10 repetitions are plotted in Figure 1.6.

The NMSE is lowest when using a rectangular grid with $m = 35$ nodes. Therefore, we choose this parameter combination for later analyses with Kohonen networks.

To use matrix factorization, we have to identify the optimal decomposition rank k first. We analyze imputation performances for the rank ranging from $k = 1$ to $k = 7$ (which is the number of columns of our matrix D) and repeat this procedure 10 times. According to the resulting NMSEs (see Figure 1.7), performance does not improve much after $k = 4$. In order to avoid overfitting the matrices W and H to the training data, we decided to use $k = 4$

Figure 1.5: Performance of k-means for different numbers of clusters

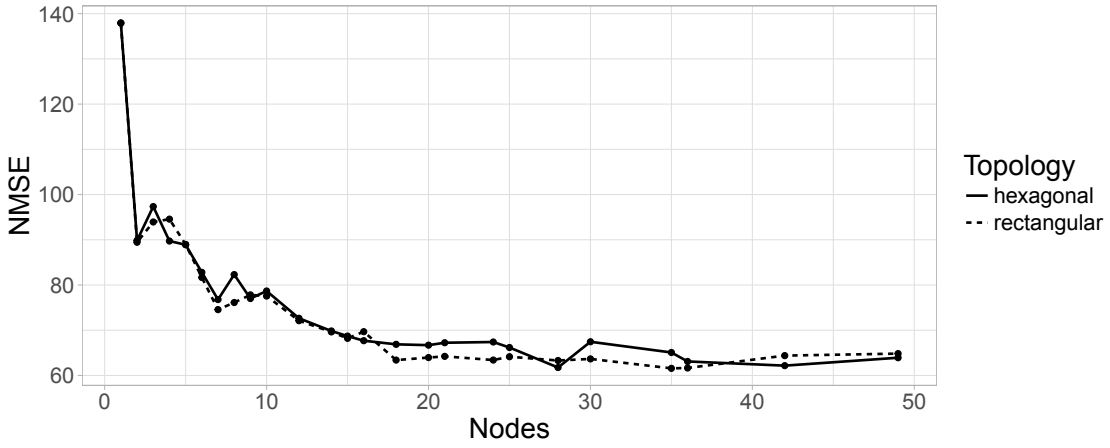


The plot shows the average performance—measured in terms of the normalized mean squared error (NMSE) stated in Equation (1.1)—of k-means for different numbers of clusters k. According to the plot, the error-minimizing k is 75, which is why we set $k = 75$ for later analyses.

when applying matrix factorization in later analyses.

Now that the optimal parameters for each model are identified, we can train the models with these parameters and finally compare the imputation performance of the models with respect to the hold-out data set. Testing on a systematic hold-out data set gives us an unbiased estimate of the error. To incorporate variations in the model training step, the models are trained five times each, resulting in five NMSEs per model. Table 1.2 shows the average NMSEs resulting from imputing the test data. As expected, the worst imputation performance results from mean imputation, with an NMSE of 186.05. Comparing the three unsupervised learning models shows that matrix factorization performs best in terms of imputation accuracy, with an NMSE of 51.08, followed by k-means, with an NMSE of 73.34. The Kohonen network model performs worst among the three unsupervised learning models, with an NMSE of 83.46. Nevertheless, the accuracies of the imputations based on models that were trained with unsupervised learning are substantially better than that of the model based on mean imputation.

Figure 1.6: Performance of the Kohonen network for different maps

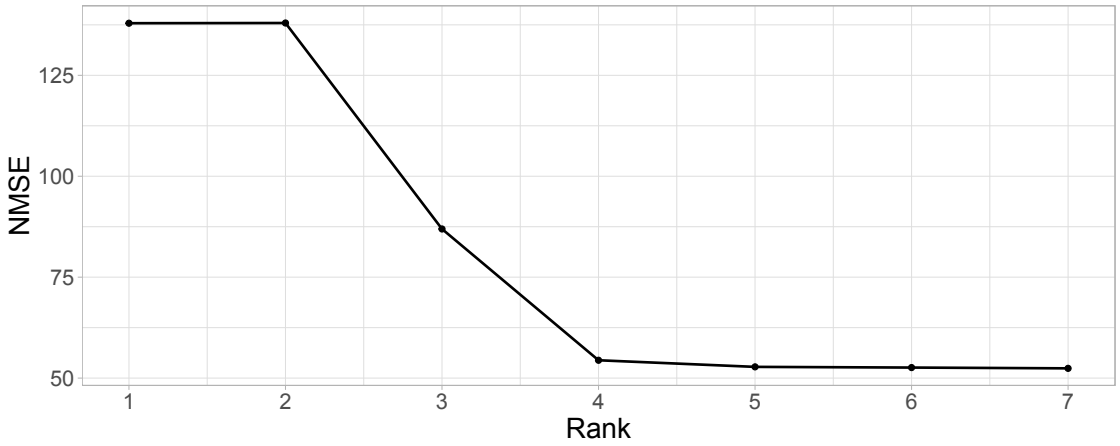


The plot shows the average performance—measured in terms of the normalized mean squared error (NMSE) stated in Equation (1.1)—of the Kohonen network for different numbers of neurons and map topologies. According to the plot, the error-minimizing map is a rectangular one with $m = 35$ nodes. Therefore, we set this parameter combination for later analyses.

Table 1.2 also shows the standard deviations of the NMSEs of the five iterations for each model, indicating the robustness of the models. It turns out that matrix factorization is the most robust model, with a standard deviation in the test-NMSE of 0. This means that a certain matrix X is always factorized to the same lower-dimensional matrices W and H, and since the product of W and H is used to predict the missing values of X, the imputation errors also do not differ from each other. In contrast, k-means is the least robust model, with a standard deviation of 8.71. As already indicated, the result of k-means heavily depends on the initial clusters that are randomly chosen as a first step when training the model.

In Section 1.4.2, we have shown that the missing values in our data set are MAR and hence that the mechanism that causes the values to be missing is ignorable. This allows us to use multiple imputation. We can therefore impute the missing values using the three unsupervised learning methods described, and repeat these imputations multiple times. As a result, we obtain multiple imputed data sets. The next step is to create probability functions from the data. Since we want to analyze the demand for apartments of different sizes, as well as the willingness to pay for apartments of specific sizes, we create probability mass

Figure 1.7: Performance of matrix factorization for different decomposition ranks



The plot shows the average performance—measured in terms of the normalized mean squared error (NMSE) stated in Equation (1.1)—of matrix factorization for different decomposition ranks k. According to the plot, the error-minimizing k is seven, which is the number of columns of our data set D. The plot also shows that performance does not improve much after $k = 4$. In order to avoid overfitting, we set $k = 4$ for later analyses.

Table 1.2: Comparison of test errors

Algorithm	**Average NMSE**	**Standard deviation NMSE**
Mean imputation	186.05	0
K-means	73.34	8.71
Kohonen network	83.46	6.91
Matrix factorization	51.08	0

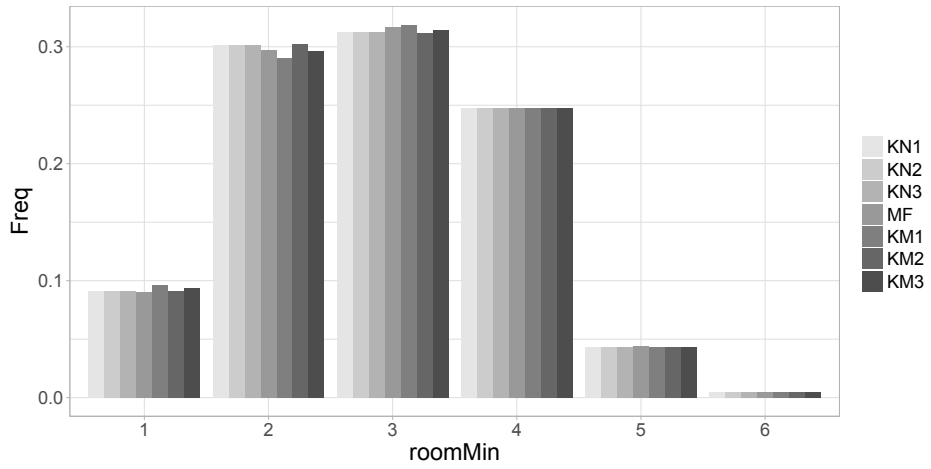
The table shows the average test errors (imputation accuracy measured in terms of the normalized mean squared error (NMSE) stated in Equation (1.1)), as well as the robustness (standard deviation of the NMSEs) of mean imputation, k-means, Kohonen network, and matrix factorization. The worst imputation performance results from mean imputation. The model is clearly beaten by each of the three unsupervised learning models. Among them, the best-performing model is matrix factorization, which is at the same time the most robust one. The second-best-performing model is k-means, which is, though, the least-robust model.

functions (PMFs) and probability density functions (PDFs) for subsamples of the data.

We restrict our further analysis to searches in the city of Zurich because the majority of the observations are searches in that city. Figure 1.8 shows the demand, determined from search subscriptions in Zurich, for apartments of different sizes. Since a person can search

in multiple areas of Zurich and since each such search is listed separately, the numbers are aggregated first (per person).

Figure 1.8: Frequency of searches for apartments of different sizes in the city of Zurich



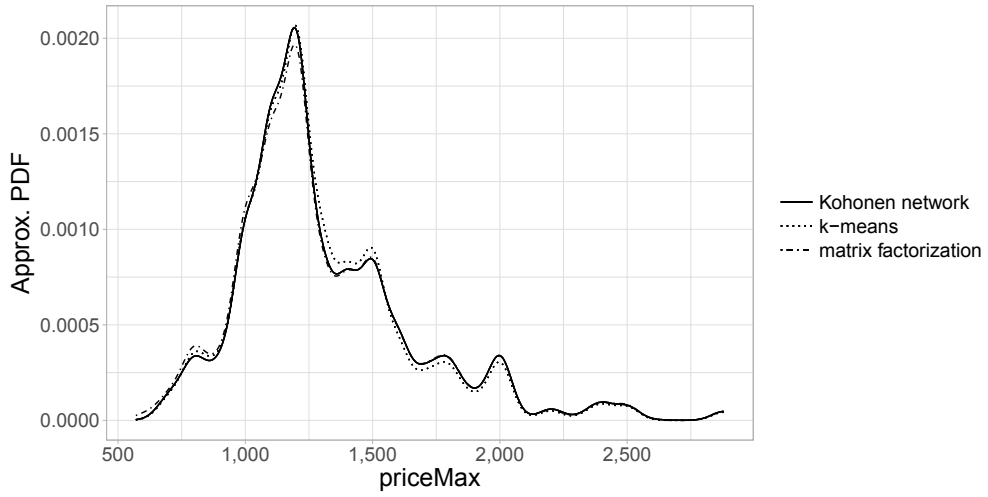
The plot shows that all the individual algorithms perform to a very similar degree. In addition, conclusions can be drawn with regard to the demand derived from search subscriptions: the demand for 3-room apartments seems to be largest, closely followed by that for 2-room apartments.

In this example, the set $\{D^0, D^1, D^2\}$ was imputed (iteratively) seven times (three times with models that are based on k-means and Kohonen networks, and once using matrix factorization[1.9]), resulting in seven completed data sets. The first conclusion that can be drawn from Figure 1.8 is that the results vary little by algorithm, showing that all the algorithms perform to a very similar degree. Second, conclusions can be drawn with regard to the demand derived from search subscriptions. For example, the demand derived from search subscriptions stipulating 3-room apartments seems to be slightly larger than that for 2-room apartments. Also, the demand for 2-room apartments seems to be larger than that for 4-room apartments. This could, however, be the result of 2-room apartments being scarcer than 4-room apartments. One could, now, go on to compare the derived demand with transaction data to see if that demand is met by supply. One could also analyze the

[1.9]We have seen that imputation with matrix factorization is robust, which means that multiply imputing the data set with matrix factorization results in equally imputed data sets. Therefore, when imputing the data set using matrix factorization, we do this only once.

willingness to pay for apartments of certain sizes. We did this for small apartments. Figure 1.9 shows seven PDFs of `priceMax` for apartments with `roomMax` ≤ 2 in Zurich.

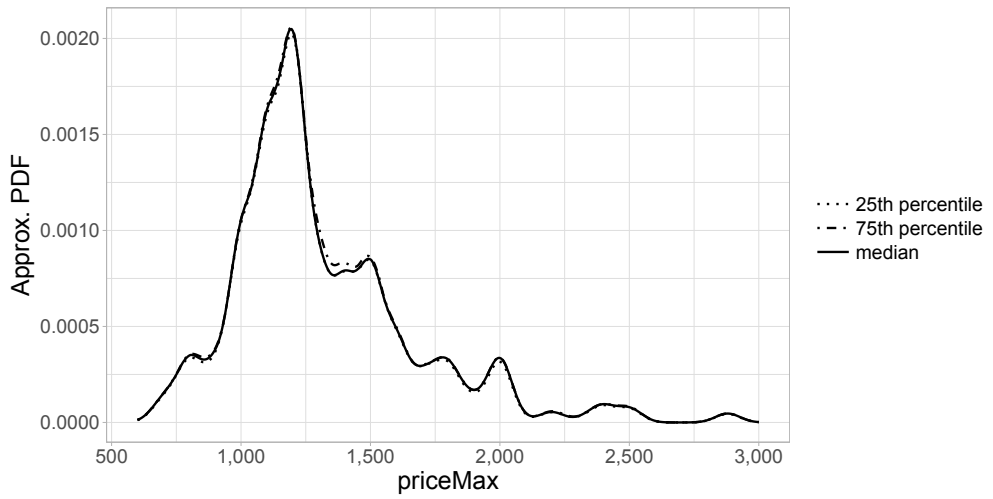
Figure 1.9: Multiple imputation outcomes



The plot shows the PDFs for small apartments in Zurich. Each line represents the PDF of one of the imputed data sets. One can see how similar the PDFs are to one another, meaning that the different imputations are very similar to one another.

In order to average over the results, there is one final step to multiple imputation: pooling. For around 2,500 prices, x, the corresponding function values, $f(x)$, are calculated. Subsequently, the usual summary statistics are computed for each x. The pooled version of Figure 1.9 can be seen in Figure 1.10. The solid line represents the median of the PDFs. The maximum willingness to pay for small apartments in Zurich peaks at 1,195 Swiss francs. The dotted and the dot-dashed lines represent the 25th and 75th percentiles, respectively. Unlike single imputation, the uncertainty of imputation is taken into account when using multiple imputation. Hence, it is possible to say by how much the maximum willingness to pay varies at any given point.

Figure 1.10: Pooled probability density functions (PDFs)



The plot shows the results from pooling, the final step of the multiple imputation process. The pooled PDFs represent the maximum willingness to pay for small apartments in Zurich. The solid line represents the median approximated function value $f(x)$ for around 2,500 prices, x. The dotted and dot-dashed lines represent the 25th and 75th percentiles, respectively. One can see how close the percentiles are to the median, showing how close the imputed data sets are to one another.

1.6 Discussion

As stated in the introduction to this paper, knowing people's preferences regarding and willingness to pay for housing is of tremendous importance. Analyses that are based on new data sources, which reveal facts about the housing market that classical transaction data cannot identify, are of great interest for many stakeholders. The example of Realmatch360 proves this point. The company analyzes data on search subscriptions to real estate platforms and sells information generated by these analyses. Launched in 2013, Realmatch360 has already acquired 140 customers, including investors, project developers, brokers, consultants, banks, and many others.

One point that invites discussion is whether what search subscriptions indicate is indeed (unmet) demand, or preferences only. As shown in Section 1.4, the maximum price stated in search subscriptions is only 6.7 percent higher than the true willingness to pay (and the majority would be willing to pay the price stated for the perfect object). Hence, people include

their budget when setting up search subscriptions, which is why we can rule out the concern that search subscriptions only express preferences. Clearly, what can thus be observed is an important indicator of future demand and hence of the development of the housing market. Rather than using transaction data alone, the use of imputed search subscriptions makes it feasible to identify what people are really searching for, and where there exists demand with no corresponding supply.

Hence, if in one region there exist only very few search subscriptions, this does not mean that there is no demand in that region. In our view, the demand for housing in that region is simply covered by the supply. It is, however, more difficult to estimate demand in regions with few search subscriptions, which is why it is of fundamental importance to use as many search subscriptions as possible, by imputing them. In addition, thanks to the data-driven method of imputation, possible changes in people's preferences can be identified.

The methods proposed in this paper clearly beat our benchmark—mean imputation. We have also shown that the imputation performances of the models that are based on unsupervised learning are very similar to one another. One has, however, to handle the algorithms with care. K-means, for example, is known to be rather unstable because it strongly depends on the initial clusters that are randomly chosen. And even though each model we trained with k-means is the result of 100 random starts, the model is still the least robust one (see Table 1.2). Therefore, it is so important to impute the data sets multiple times and average over their values in the pooling step.

1.7 Conclusion

As is the case for many data sets exhibiting missing values, imputation is of particular importance—in our case of looking at data on search subscriptions to housing platforms, ignoring all observations where there are missing values would lead to a completely distorted picture of unmet demand. We have seen that our three models that are trained using unsupervised learning perform similarly well, even though the approaches are very different.

In addition, they clearly beat our benchmark—mean imputation. The most accurate results were achieved by averaging over the models that were trained with unsupervised learning algorithms.

Compared to the use of transaction data alone, the use of imputed search subscriptions allows one to identify what people are really searching for and where there is unmet demand. This may serve to preclude additional accommodation being built in regions where there is no unmet demand; to improve the pricing of accommodation thanks to enhanced knowledge of the willingness to pay for specific objects in different regions; and even to identify regions where there is unexpected demand.

Further analyses could include attempts to understand what drives demand. It would also be interesting to see the results from the analyses described here if more features would be available (such as if accommodation has a balcony, or which floor an apartment is on).

Even though this research was conducted on housing data from Switzerland, the methodological insights can be extended to other countries or even to other markets.

Appendix A

Pseudocode of the k-means algorithm used

Algorithm 1 Iterative imputation using k-means based on Lloyd (1982)

Look at the following subset: `searchType`=rent∧`objectType`=apartment.
Drop observations where
    is.empty(`priceMin`)∧is.empty(`priceMax`) or
    is.empty(`roomMin`)∧is.empty(`roomMax`).
for $m = 1, \ldots, p-2$ **do**
    Training set$_{(n_1 \times p)}$ = all observations with 0 missing values.
    Target set$_{(n_2 \times p)}$ = all observations with m missing value(s).
    Among training set: randomly initialize k cluster centroids
    $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^p$ (p = number of features).
    repeat
        for $i = 1, \ldots, n_1$ **set**
            $c^{(i)} := \arg\min_{j \in [1,k]} \text{dist}(x^{(i)}, \mu_j)$.
        end for
        for $j = 1, \ldots, k$ **set**
            $\mu_j := \frac{\sum_{i=1}^{n_1} 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{n_1} 1\{c^{(i)}=j\}}$.
        end for
    until convergence
    for $i = 1, \ldots, n_2$ among target set **do**
        Find $j^* = \arg\min_{j \in [1,k]} \text{dist}(x^{(i)}, \mu_j)$.
        Substitute missing feature(s) of $x^{(i)}$ by corresponding feature(s) of μ_{j^*}.
    end for
end for

Appendix B

Pseudocode of the Kohonen network algorithm used

Algorithm 2 Iterative imputation using the Kohonen network by Kohonen (1982)

Look at the following subset: `searchType`=rent∧`objectType`=apartment.
Drop observations where
    is.empty(`priceMin`)∧is.empty(`priceMax`) or
    is.empty(`roomMin`)∧is.empty(`roomMax`).
for $m = 1, \ldots, p-2$ **do**
    Training set$_{(n_1 \times p)}$ = all observations with 0 missing values.
    Target set$_{(n_2 \times p)}$ = all observations with m missing value(s).
    Define the topology of the map.
    Randomly initialize the map's nodes' weight vectors w.
    repeat
        Grab an input vector $x^{(i)}$.
        Compute similarity between $x^{(i)}$ and each node v's weight vector w_v.
        Find closest node, u.
        Update the nodes' weight vectors:
            $w_v(s+1) = w(s) + \theta_{uv}(s)\alpha(s)(x^{(i)} - w_v(s))$,
            where $\theta_{uv}(s)$ is a neighborhood function converging to 0 as dist$(u, v)\uparrow$,
            and α denotes the learning rate.
    until maximum step size reached
    for $i = 1, \ldots, n_2$ among target set **do**
        Find $v^* = \arg\min_v \text{dist}(x^{(i)}, w_v)$.
        Substitute missing feature(s) of $x^{(i)}$ by corresponding feature(s) of
        w_{v^*}.
    end for
end for

Appendix C

Pseudocode of the matrix factorization (MF) algorithm used

Algorithm 3 Iterative imputation using MF based on Lee and Seung (1999, 2001)

Look at the following subset: `searchType`=rent∧`objectType`=apartment.
Drop observations where
    is.empty(`priceMin`)∧is.empty(`priceMax`) or
    is.empty(`roomMin`)∧is.empty(`roomMax`).
for $m = 1, \ldots, p-2$ **do**
    Data set $V_{(n\times p)}$ = all observations with $\leq m$ missing value(s).
    Randomly initialize $W_{(n\times k)}$ and $H_{(k\times p)}$.
    $\{W^*, H^*\} = \arg\min_{W,H} \text{dist}(V, WH)$,
    where W^* and H^* can for example be found using Lee and Seung (2001)'s multiplicative update rule.
    Update V: $\tilde{V} = W^*H^*$ (no missing values any longer)
end for

Recommender Systems for Brick-and-Mortar Travel Agencies[2.1]

Vanessa Kummer, University of Zurich

Abstract

Brick-and-mortar travel agencies have one big advantage over online travel agencies: the personal relationships they maintain with their customers and the resulting knowledge that agents have about those customers. However, ever-increasing competition in the form of online offers that enable travelers to book tours themselves is forcing classical travel agencies to reconsider their strategies. Travel agencies have collected a lot of data about their customers over recent years and by now they have all realized that making use of it would be a tremendous advantage; a large majority of travel agencies, however, still do not know how to do so. With this in mind, this paper provides a manual on how to set up recommender systems as a supportive tool based on customers' travel histories. Recommender systems can help travel agencies to provide better services and ultimately increase their revenues. The paper provides detailed information on two approaches to collaborative filtering—neighborhood models and matrix factorization—and compares their performance with that of the popularity model and of a recommender suggesting customers' most recently booked trip. In addition, the paper shows how including user and item metadata, as well as predicting customers' ratings by using implicit feedback data, leads to a sophisticated but still easy to set up recommender system.

[2.1]I would like to thank D-rt Groep B.V. for providing me with data, and Jan Henne De Dijn from D-rt Groep B.V. for his thoughtful support. I am also grateful to René Algesheimer, Rob Earle, Felix Kübler, Harry Paarsch, Karl Schmedders, and seminar participants at the University of Zurich for their feedback on my work.
I gratefully acknowledge the financial support of the University of Zurich (Forschungskredit of the University of Zurich).

2.1 Introduction

Travel and tourism is one of the world's largest economic sectors. In its annual analysis of the global economic impact of travel and tourism, the World Travel & Tourism Council states that the sector accounted for 10.4 percent of global GDP and 313 million jobs, or 9.9 percent of total employment, in 2017 (World Travel & Tourism Council, 2018). Ongoing innovations will continue to drive growth and change across the sector. Between 2016 and 2018, travel start-ups cumulatively raised USD 30 billion in funding—almost the sum raised over the previous decade (Weissenberg and Langford, 2018).

These innovations have led to a market share shift from classical, "offline" travel agencies to online ones. According to Messe Berlin GmbH, which yearly publishes the *ITB World Travel Trends Report*, online bookings reached about 65 percent of total bookings in 2015, while travel agency bookings dropped to about one-fifth in Europe (Messe Berlin GmbH, 2015). Traditional travel agencies are struggling to keep their customers due to perceived higher prices and a smaller availability of options compared to online travel agencies (OTAs). One important benefit that traditional travel agencies have compared to OTAs, though, is the personal relationships they maintain with their customers. These personal relationships play a vital role in influencing buyer perceptions of expertise, trust, and relationship loyalty (Newell et al., 2011). Committed relationships are a sustainable advantage for sellers as they are difficult for competitors "to understand, to copy or to displace" (Day, 2000). Ultimately, loyal customers lead to an increase in market share and revenues, and to a decrease in costs related to acquiring and maintaining customers (Reichheld, 1993). These personal relationships are hence of clear benefit for travel agencies and should be maintained. What can classical travel agencies do then to avoid losing further market share to OTAs? The core of OTAs is their know-how on how to make use of the data they have. While traditional travel agencies also have a lot of data at hand, they often still lack knowledge of how to use this data. The question therefore is obvious: How can offline agencies make use of their data without losing their advantage of personal relationships?

This paper suggests the introduction of recommender systems (RSs) as a supportive tool for travel agents. RSs are well known, for example, in the movie industry (Koren et al., 2009) or in online retailing (Linden et al., 2003). To the best of our knowledge, the present paper is the first study that shows how RSs—systems that seek to predict, on the basis of data on user–item interactions, the rating or preference a user would give to an item—can help brick-and-mortar travel agencies to improve their businesses. Untypically for RSs though, the RSs proposed in this paper are not intended to be used by the "users" of the RS—here referred to as "customers"—but by (in this case) intermediary travel agents.

The models are trained on past travel data and can—once trained—be used to predict travel preferences. The performance of our RSs is compared to that of popularity models and of a recommender that simply suggests customers' most recently booked trip. We have chosen these two models as baselines, because—according to Jan Henne De Dijn, Managing Director at D-rt Groep B.V.—travel agents often recommend to their customers either destinations that have been booked previously or trending destinations. We will see that the first of these approaches is a difficult one to beat but that it has some drawbacks. The latter approach (recommending the most popular regions), on the other hand, can be beaten by RSs thanks to their ability to personalize. Not only does the use of a well-designed RS increase the number of items sold, it also increases user satisfaction (Ricci et al., 2011). Thus, a good RS can improve customers' experience with a travel agency.

To stress the importance of using such a system, consider the following situation: As of spring 2018, bookings with the two Dutch travel agencies D-reizen and VakantieXperts started to stagnate. This is per se nothing too unusual as early bookers have already booked their trips by this time, while the more spontaneous customers wait until the holiday season approaches. The big problem that many travel agencies—including D-reizen and VakantieXperts—faced in summer 2018, though, was that due to the nice weather in Europe people did not consider booking vacations and so booking numbers dropped drastically. Sales agents from the two travel agencies therefore started a campaign of contacting leads (customers who had already

booked a trip at the same time of the previous year) and providing them with holiday proposals. Proposals consisted of trips the customers had booked in the past or of best sellers (interview with Jan Henne De Dijn, Managing Director at D-rt Groep B.V., 8 November 2018).

The individual travel agents certainly know their customers and their needs and interests, and are therefore able to suggest holidays that fit these customers very well. But in the case of such a campaign this knowledge does not come into play. Due to the campaign's large scale, call center agents were instructed to call customers even though they did not know them personally. RSs could have served as a great supportive tool, and been used to personalize holiday proposals. And, of course, there exist many more situations in which an RS would be of great help: one could, for example, personalize email newsletters, or send catalogs according to customers' interests. There exist many such examples of where agents' knowledge of their customers cannot be used due to largeness of scale; in these situations, an RS is a tremendous advantage. In addition to reducing costs, adopting RSs where personal interaction is missing leads to customers feeling understood due to the degree of personalization. And feeling understood, in turn, leads to an increased customer satisfaction that, ultimately, strengthens agencies' relationship with customers (Buttle, 2004).

In 2000 already, Buhalis published a paper in which he stated that technologically advanced tourism organizations will enhance their competitiveness in the future, while those who fail to take advantage of technological developments will lose market share and eventually be driven out of the market (Buhalis, 2000). But according to a recently published article by Weissenberg and Langford (2018), technology alone will not give brands all of the tools they need if they are to succeed. The authors state that too much focus on technology has the potential to create cold and robotic experiences and environments, when travel is still very much a people-to-people experience. Instead, travel brands should focus on leveraging technology to produce elevated, authentic experiences without losing sight of the human connection. The authors state that, ultimately, "for travel brands, people and culture will

always be a competitive advantage" (Weissenberg and Langford, 2018). This is the big advantage that classical travel agencies have compared to OTAs. To further enhance customer experience, travel agencies should work on gaining insights into their data. According to *The State of Data and Analytics in Travel Report 2017* (2017), only 65 percent of travel businesses currently have a dedicated data analysis team. In general, simpler types of analytics are adopted: according to the survey that The State of Data and Analytics in Travel carried out in 2017, the most common form of analytics is *diagnostic* (trying to explain why information is as it is, including by finding correlations), followed by *descriptive* (summarizing what has happened or is happening currently, usually in the form of simple metrics). The State of Data and Analytics in Travel also shows that the majority of travel businesses are not yet able to use their data to build *prescriptive* analytics: using modeling and forecasting to discover and project results derived from past patterns in data (The State of Data and Analytics in Travel, 2017). This problem is tackled in this paper by showing how RSs can be set up and deployed by travel agencies.

The remainder of the paper is organized as follows: In Section 2.2 we take a look at what RSs are and how they can be set up. Section 2.3 introduces the data used and the methods applied in this paper. The results of the analyses performed are presented in Section 2.4, and a discussion of the results follows, in Section 2.5. Section 2.6 concludes.

2.2 Recommender Systems

RSs have become increasingly popular. One of the most famous examples is certainly Amazon's RS, which recommends products to customers based on the purchasing decisions made by similar customers. Another example is Netflix, which uses an RS based on ratings to decide which movies and TV shows to present to its customers. But RSs are also utilized in a variety of other areas, including news, search queries, financial services, and online dating. In this section we provide a short summary of RSs.

The basic idea of RSs is to utilize various sources of data to infer a person's interests. The entity to which the recommendation is provided is referred to as the user and the product being recommended is referred to as an item. Hu et al. (2008) distinguish between two different techniques (and combinations thereof) for setting up RSs: the content-based approach and collaborative filtering (CF). Content-based methods create profiles for each user and each item, aiming to characterize their nature. The resulting user and item profiles are then matched in order to generate recommendations. Items that are recommended have similar profiles to that of the user to whom they are recommended. This method requires collecting information on items in order to characterize them, although such information may not be available.

CF, on the other hand, only relies on past user–item interactions. From those interactions, it aims to find relationships between users and items in order to identify users' preferences for new items. While generally being more accurate than content-based techniques, CF suffers from the cold-start problem (Hu et al., 2008).

It should be noted that there also exist further types of RS, such as demographic, utility-based, or knowledge-based RSs (Burke, 2002). CF is probably the most widely implemented variety though. The most common approach to CF is based on neighborhood models. A more holistic approach to CF is latent factor models, including matrix factorization, which aim to uncover latent features that explain ratings. Recently, matrix factorization techniques have gained popularity thanks to their superior scalability (Mnih and Salakhutdinov, 2008). The two approaches—neighborhood models and matrix factorization—will be explained in greater detail in the following two sections.

2.2.1 Neighborhood Models

CF is based on previous interactions between users and items. Data on these interactions is stored in so-called utility or rating matrices that indicate, for each user–item pair, a value that represents the degree of preference of a specific user for a particular item. An example

of a utility matrix is shown in Table 2.1. The problem is that some (or in practice many) of these ratings are missing, implying that there is no information about some users' preference for that item. The goal is to predict these missing ratings. Neighborhood models are based

Table 2.1: Example of a rating matrix

	Item 1	**Item 2**	**Item 3**
User 1	4	5	1
User 2	2		3
User 3		3	2

This is an example of how a utility matrix—the foundation for collaborative filtering—looks. Each row represents a user, and each column an item. The numbers represent the users' ratings for the items. In this example, the ratings are numbers from one to five, but often we only have either 1s (indicating that the user, for example, *has* bought the item or *has* watched the movie) or blanks—in the latter case the ratings are then referred to as *unary*.

on the similarity between users and/or items. In the user–user approach, the rating of user u for item $i = 1, \ldots, m$ is predicted by averaging the ratings for item i of the k most similar users to user u. The similarity between two users, u and v, is typically measured by the cosine similarity

$$sim_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_i (r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_i (r_{v,i} - \bar{r}_v)^2}}, \tag{2.1}$$

where I is the set of items rated by both user u and user v, $r_{u,i}$ is the rating of user u for item i, and $\bar{r}_u$ is the average rating of user u (and equivalently for user v).

Applying Equation (2.1) to Table 2.1 implies a similarity between user 1 and user 2 of $sim_{1,2} = -0.72$, and a similarity between user 1 and user 3 of $sim_{1,3} = 0.96$.[2.2] The reason for user 1 and user 3 being more similar to each other than user 1 and user 2 is that user 1 and user 3's ratings for the items that they have rated in common have the same ranking, whereas for user 1 and user 2 the rankings are reversed.

[2.2] $\bar{r}_1 = \frac{4+5+1}{3} = 3.3$, $\bar{r}_2 = \bar{r}_3 = \frac{2+3}{2} = 2.5$,

$sim_{1,2} = \frac{(4-3.3)(2-2.5)+(1-3.3)(3-2.5)}{\sqrt{(4-3.3)^2+(5-3.3)^2+(1-3.3)^2}\sqrt{(2-2.5)^2+(3-2.5)^2}} = -0.72$,

$sim_{1,3} = \frac{(5-3.3)(3-2.5)+(1-3.3)(2-2.5)}{\sqrt{(4-3.3)^2+(5-3.3)^2+(1-3.3)^2}\sqrt{(3-2.5)^2+(2-2.5)^2}} = 0.96$.

Dually, item similarity can be used to estimate the rating of user u for item i by finding the k items most similar to i and taking the average rating that u has given to those k items. Alternatively, we can also look at all m items and weight their ratings with their similarity to item i. In theory, user–user and item–item CF should perform similarly well, but in practice item–item CF often provides more reliable information because it is easier to discover items that are similar than to detect that two users are similar: two users might like the same item while each also likes another item that the other does not care for. In that sense, items are "simpler"/more consistent than users. In addition, as the number of users increases, user-based CF runs into problems with scalability whereas item-based CF does not (Aggarwal, 2016; Leskovec et al., 2014; Sarwar et al., 2001).

2.2.2 Matrix Factorization

An entirely different approach to estimating the blank entries in the utility matrix is to conjecture that the utility matrix is actually the product of two lower-dimensionality matrices. Matrix factorization is an example of such a dimensionality reduction algorithm. Its goal is to approximate a matrix $M \in \mathbb{R}^{n \times m}$ (in our context, M equals the utility matrix with the missing ratings) by searching for two lower-dimensional matrices, $U \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{d \times m}$, such that the product UV closely approximates the non-blank entries of M (see Equation (2.2)). One approach to deriving U and V is to apply stochastic gradient descent. As the objective function contains $m \times n$ terms though, a better approach to deriving U and V if $m \times n$ is too large is to use alternating least squares (Hu et al., 2008). The blank entries of M are then predicted by the product UV.

$$\underbrace{\begin{bmatrix} 4 & 5 & 1 \\ 2 & & 3 \\ & 3 & 2 \end{bmatrix}}_{M} \approx \underbrace{\begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \end{bmatrix}}_{U} \times \underbrace{\begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \end{bmatrix}}_{V}. \tag{2.2}$$

2.3 Data and Method

To train our RSs, past travel data from the Dutch company D-rt Groep B.V. is used. D-rt Groep B.V. consists of the travel agencies D-reizen and VakantieXperts, and is the retail market leader in the Netherlands, with a total of 1,800 employees and 450 shops, and various websites (D-rt Groep B.V. (D-reizen & VakantieXperts), 2018). Unfortunately, its travel data does not contain any explicit ratings from its customers with regard to the trips they have booked, only the fact that they *have* booked them. This is the struggle that many real-life applications of this kind imply. One big challenge that comes with the fact of missing explicit ratings is that there is no information on the negative preferences of customers (Baltrunas and Amatriain, 2009).

We therefore start with the approach of using these so-called unary ratings. In a second step, customers' ratings are predicted by using implicit feedback data.

There is a huge literature on how to use implicit data for RSs. Hu et al. (2008) base their RS on the measure of confidence with regard to whether a user would like or dislike an item ("preference"), both confidence and preference being derived from the input data. In contrast to Hu et al. (2008), who optimize their model to predict if an item is selected by a user or not, Rendle et al. (2009) base the optimization of their model parameters on item pairs and these item pairs' *ranks* for each user. Yi et al. (2014) go one step beyond binary implicit feedback to investigate the interactions between users and items. They use the amount of time that a user spends on content items ("dwell time") as a proxy to quantify the likelihood with which a content item is relevant to a particular user. The approach we use is similar to that of Hu et al. (2008): we approximate a customer's rating for a category by counting the number of trips that customers booked per category. More trips booked in one category translates into a higher rating for that category. Since the ratings are normalized, a customer's rating is relative to the total number of bookings that that customer made.

In the context of travel agencies, the level of specificity of recommended items can range from single hotels to countries, or even to continents. Since, ultimately, travel agents need

to obtain information from the RS on where customers would probably want to travel to, but at the same time the user–item matrix should not be too sparse, we consider *regions* the customers traveled to as items/categories. Incorporating only customers who have booked trips in more than two regions results in a rating matrix of dimensions of around $n = 400,000$ and $m = 1,000$. The average customer has traveled to 3.14 regions, resulting in a sparsity of the user–item matrix of 99.70 percent. This means that the average customer has not even traveled to 0.5 percent of regions.

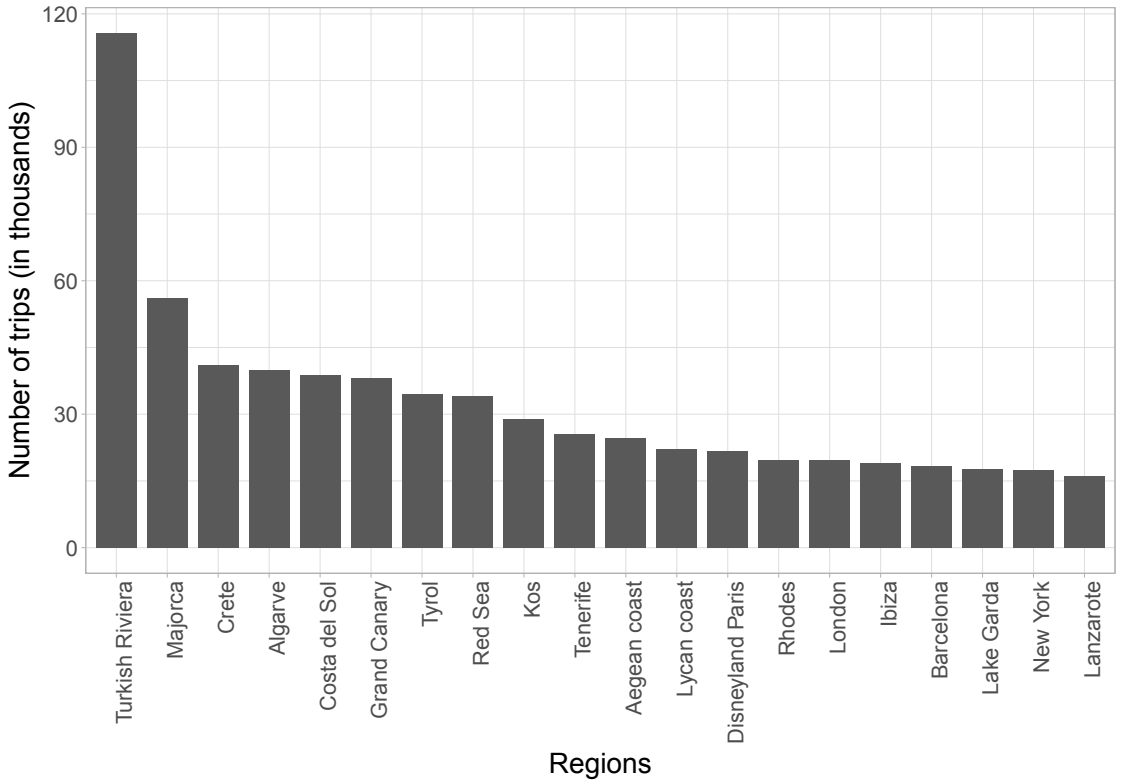
Figure 2.1 shows the most-booked regions according to the data set. The plot shows that some regions in the data set, such as the Turkish Riviera—referring to a whole area of southwest Turkey—are rather broadly specified, whereas, for example, Disneyland Paris or Ibiza are quiet specific. In addition, the plot shows that the most popular regions are warm and sunny regions. This also explains why the nice weather in Europe in summer 2018 led to such a drastic drop in booking numbers.

To measure how well the models perform, in a first step 10 percent of the customers are randomly selected. From those customers' item vectors, the most recently booked trip is removed and stored for later comparison with the recommended region. These out-of-sample trips are only used at the very end to compute the testing error.

In a second step, another 10 percent of the customers are randomly selected, and from each of those customers' item vectors one booked trip is removed and stored for later validation. This second step of the validation set approach is repeated ten times, and in every iteration multiple models are trained and the top k recommended regions for each user are filtered. These k recommendations are then compared to the validation regions.

Obviously, the higher is k, the bigger are the chances that the recommendations include the previously removed trip. If we would recommend *all* our items to customers, the list of recommendations would certainly include items that our customers like. But this does not mean we have a good model. A model that recommends to a customer ten items of which the customer likes three is more helpful than a model that recommends 100 items of

Figure 2.1: Best sellers



This plot shows the regions most often booked by D-rt Groep B.V.'s customers. Some, such as the Turkish Riviera—referring to a whole area of southwest Turkey—are rather broadly specified, whereas, for example, Disneyland Paris or Ibiza are quite specific. The most popular regions are apparently warm and sunny, which would also explain why the nice weather in Europe in summer 2018 led to such a drastic drop in booking numbers. Data source (for this and all subsequent figures and tables): D-rt Groep B.V.

which the customer only likes three. Therefore, we compute not only the probability that a relevant item is recommended, which is referred to as *recall at k*,

$$recall\ at\ k = \frac{\#\ hits}{\#\ relevant\ items}, \tag{2.3}$$

but also the proportion of recommended items in the top k set that is relevant, called *precision at k*,

$$precision\ at\ k = \frac{\#\ hits}{k}. \tag{2.4}$$

Precision and recall are the most common metrics used for evaluating information retrieval

systems. In 1966, Cleverdon and Keen proposed them as the key metrics, and they have been used ever since. The higher the two measures are, the better the model's performance. Since there exists a trade-off between precision and recall, we can combine the two measures by looking at the *F-measure*:

$$F = \frac{2 \times precision \times recall}{precision + recall}. \tag{2.5}$$

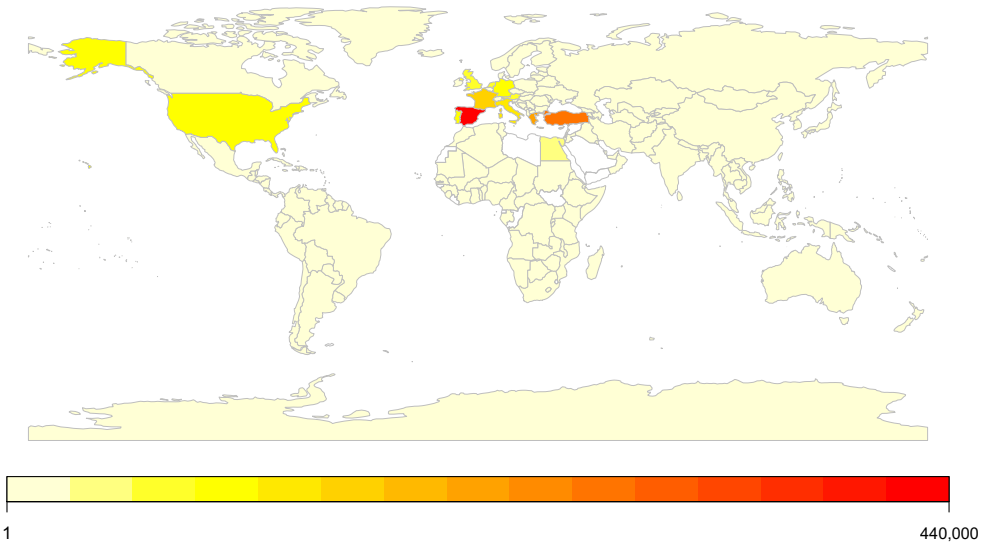
Ultimately, the F-measure also helps us to identify the optimal list length k that balances out precision and recall best.

2.4 Results

We introduce two baseline models that our RSs should optimally beat: one is a system that simply recommends the most recently booked trip of the respective customer, and the other is a so-called popularity-based RS. A popularity model ranks items by the number of times they were purchased and recommends them, in the order of their rank, to users. Looking at Figure 2.2, this approach seems to be a difficult one to beat because the customers from our data set are very homogeneous. We see that by far the majority of D-rt Groep B.V.'s customers travel to Spain, followed by Turkey, Greece, France, and Italy.

As Johan Ydring, Head of Brand and Performance Marketing at TUI Nordic, states in a book by Sumpter, when it comes to marketing to specific target groups "Often the simplest ideas are best" (Sumpter, 2018). We can therefore expect beating these two baselines to be challenging.

In the following, the performance of RSs trained using neighborhood models and using matrix factorization is compared to that of our baseline models. Then, it is shown how including user and item metadata changes the performances of the RSs that are trained on that data. In a last step, we will see how predicting ratings by implicit feedback data, and training RSs on those predicted ratings rather than on unary ratings, affects the performance of the RSs.



This heat map shows how homogeneous the customers of D-rt Groep B.V. are. By far the majority travel to Spain, followed by Turkey, Greece, France, and Italy.

2.4.1 Basic Models

To start with, consider the campaign outlined in the introduction: sales agents call leads and propose them vacation destinations. In such a case, they most probably only recommend one or two trips to prevent overwhelming the customer. The average probability that the suggested trip is of interest to the customer (measured in terms of recall at $k = 1$ and $k = 2$ from Equation (2.3)) when recommending best-selling trips is 6.50 percent for $k = 1$ and 10.61 percent for $k = 2$. Recommending the most recently booked trip results in a recall rate of 16.16 percent (see Table 2.2). It is not surprising that the latter baseline model performs more than twice as well as the former: the latter is personalized while the former is not. However, the latter, most recently booked trip baseline model will never recommend regions that are new to a customer and the model's added value is therefore not very high for travel agents.

Let us now check how the proposed RSs would perform in such a situation. The first RS is trained using item–item CF. As proposed by Deshpande and Karypis (2004), we normalize

each customer u's item vector x to be of unit length: $x' = \frac{x}{||x||}$. The resulting normalized ratings $\mathbf{r}_u \in \mathbb{R}^m$ (blanks being replaced by zeros and m being the number of regions) are then used to compute the score vector.

The first step is to calculate the item–item (region–region) similarity based on each customer's normalized rating vector r_u. Then, we predict the likelihood of booking a trip in a certain region by calculating the similarity between the target region and regions that the customer of interest has already booked a trip to. The weighing factor would be the rating of the respective regions. This weighted sum is then scaled by dividing by the sum of the similarity measures of the target region. The result is a vector of scores for all regions, which we can sort in order to identify the top k regions for the customer of interest u:

$$\mathbf{S}_u = (\mathbf{W}^T \mathbf{r}_u) \begin{pmatrix} \frac{1}{\mathbf{w}_1} \\ \vdots \\ \frac{1}{\mathbf{w}_m} \end{pmatrix}, \tag{2.6}$$

with $\mathbf{W} \in \mathbb{R}^{m \times m}$ being the item similarity matrix and $\mathbf{w}_1, \ldots, \mathbf{w}_m$ the column sums of $\mathbf{W}$ (Sarwar et al., 2001; Köhler, 2017).

After normalizing all rows of Table 2.1, our exemplary rating matrix, to unit length, the cosine similarity matrix $\mathbf{W}$ would look as follows:[2.3]

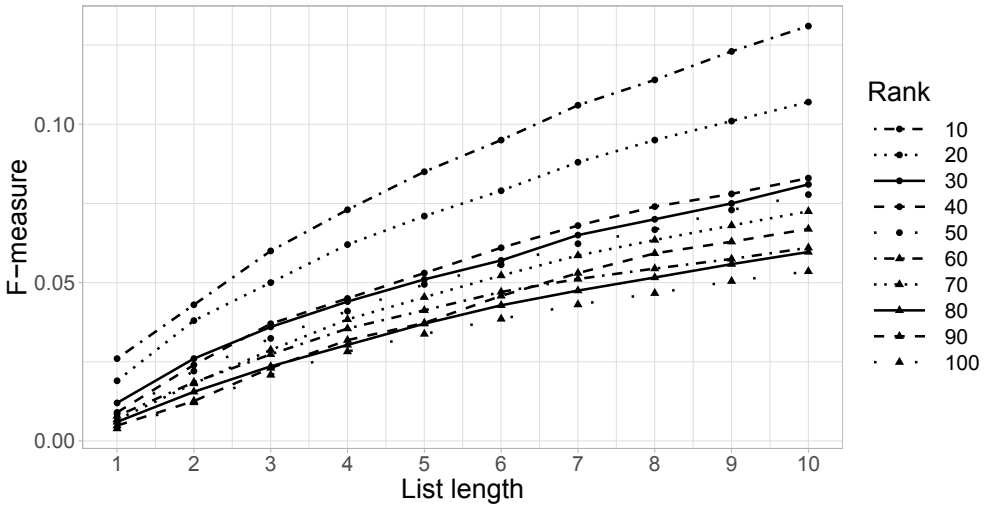
$$\mathbf{W} = \begin{pmatrix} 1 & 0.47 & 0.73 \\ 0.47 & 1 & 0.46 \\ 0.73 & 0.46 & 1 \end{pmatrix}.$$

The resulting column-sums vector would then be $(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = (2.20, 1.93, 2.19)$, leading to a score vector for user 2 of $\mathbf{S}_2 = (0.53, 0.33, 0.56)$. Recommendations for user 2 would then be based on the values of $\mathbf{S}_2$.

[2.3] Note: the similarities in $\mathbf{W}$ are different from the ones derived in Section 2.2.1 because the normalization technique differs.

The second RS is trained using the previously explained matrix factorization algorithm. A comparison of the model's performance in the validation step for different ranks d is shown in Figure 2.3. The plot shows that—for RSs based on unary ratings—a decomposition rank of $d = 10$ is optimal.

Figure 2.3: F-measure values of unary ratings-based RSs trained using matrix factorization at $k = 1, \ldots, 10$ for different decomposition ranks d



This plot shows the validation performance of unary ratings-based recommender systems trained using matrix factorization. According to our analyses, a decomposition rank of $d = 10$ performs best (in terms of the F-measure as stated in Equation (2.5)) for all list lengths analyzed.

The resulting out-of-sample recall rates of the two models at $k = 1$ and $k = 2$ are shown in Table 2.2. Both RSs indeed outperform the popularity-based model at $k = 1$ as well as at $k = 2$, even though we have not added any additional data.[2.4] However, both RSs are outperformed by the second baseline model—recommending customers' most recently booked trip. This might however be the reflection in the data of how travel agents have recommended trips to their customers in the past. In addition, the big advantage of RSs is that they are able to identify items that users are not yet aware of, but would probably like. Recommending customers' most recently booked trip, however, only recommends items that the customer already knows, resulting—as previously mentioned—in little added value for

[2.4]Comparison of the models with respect to the F-measure is discussed in the next section.

the travel agent.

Table 2.2: Out-of-sample recall rates of unary ratings-based RSs at $k = 1$ and $k = 2$

Model	**Recall at** $k = 1$	**Recall at** $k = 2$
Popularity-based	0.065	0.106
Most recently booked	0.162	
Item–item CF	0.133	0.187
Matrix factorization	0.123	0.164

This table shows that the popularity-based model is outperformed (in terms of *recall at* k as stated in Equation (2.3)) by both basic recommender systems (RSs) for $k = 1$ as well as $k = 2$. However, the baseline model recommending customers' most recently booked trip outperforms the basic RSs. This might however be the reflection in the data of how travel agents have recommended trips to their customers in the past.

2.4.2 User and Item Metadata

The methods explained in the previous section can be extended by including so-called user and item metadata. Companies such as Amazon have information about their customers on the individual level, which they can include to train their RSs. Unfortunately, our data set only contains little such information. Hence, on the individual level, we only include information about age, gender, and nationality, as well as the place of residence of the customers. In addition, we have created a dummy variable, indicating whether a customer has children under the age of 15.

To include more information on users in the training data, we add aggregated information on the customers' place of residence. Adding data from external sources is also suggested by Buttle (Buttle, 2004). Hence, from the Centraal Bureau voor de Statistiek, we include the following data for the year 2017 on the customers' place of residence:

- Urbanity
- Population size
- Number of foreign residents
- Number of single-person + childless, multi-person + multi-person households

- Average household size
- Number of cars

In order to be able to find correlations between users and the types of regions they like, we also include metadata on regions:

- Continent
- Country
- Whether Dutch is a national language of that country
- Crime rate
- GDP per capita (as an approximation for the price level)

To make use of user and item metadata, a hybrid approach is adopted. Hybrid RSs use information from both user–item interactions and users' as well as items' characteristics. Because of this combination, hybrid methods are less prone to the cold-start problem—which arises when a user or item has no or little activity—than RSs that are only based on the rating matrix. In addition, because of the augmented data base, hybrid RSs also suffer less from the sparsity problem (Adomavicius and Tuzhilin, 2005). Another advantage of hybrid RSs is that multiple RSs—each of them, individually, having certain drawbacks—can be combined such that the resulting hybrid RS no longer suffers from those drawbacks (see for example Gurbanov and Ricci (2017), who combine CF with sequence mining).
RSs can be combined in a variety of ways. The recommendation scores of various RSs can, for example, be combined to a single score (weighted hybridization method), features from different recommendation data sources can be merged and fed into a single algorithm (feature combination), or the output from one technique can even be used as an input for another (feature augmentation) (Burke, 2002).
In addition to CF, we therefore introduce a content-based approach and a demographic one. As already explained, content-based methods make recommendations by comparing descriptions of items that have been rated to descriptions of items to be recommended.

Demographic models use users' demographic information to identify what types of user like what items (Pazzani, 1999).

To find regularities among the descriptions of regions liked by a given user, we first compute each user's profile by taking the dot product of their ratings and the item metadata (scaled by subtracting the variable mean and then dividing by the variable's standard deviation). In a second step, a score for each user–item pair is computed by calculating the similarity between items' characteristics and users' profiles. Recommendations are then based on these scores.

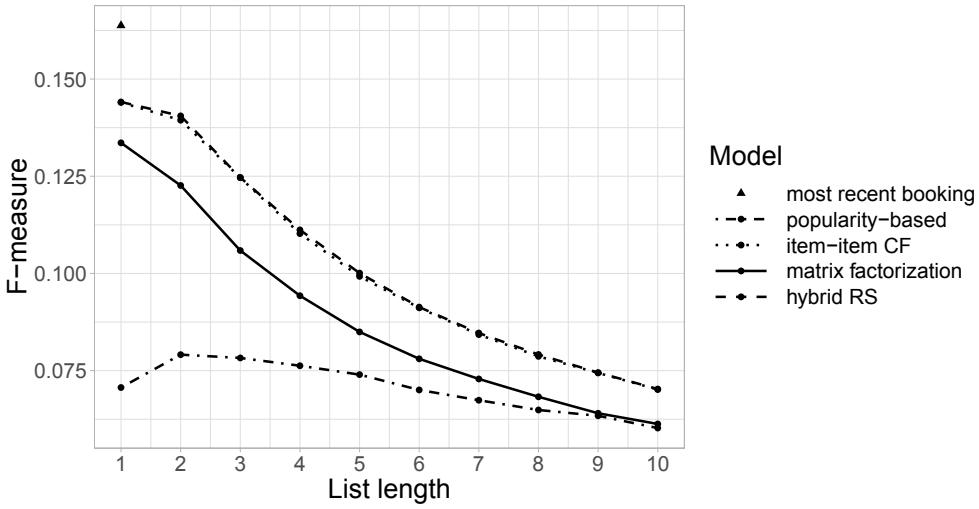
Similarly, we can find regularities among the descriptions of users that like certain regions. We therefore take the dot product of the user–metadata matrix (scaled in the same way as the item–metadata matrix) and users' ratings. This results in a matrix that indicates the extent to which users with certain metadata like each of the regions. In order to then identify a score for each user–item pair, we simply calculate the similarity of each user's metadata to the previously obtained matrix.

In order to combine the three RSs, we apply weighted hybridization, meaning that we linearly combine the three score matrices. The weights are determined by running the same validation set approach as the one used for identifying the optimal rank in matrix factorization.

Taking a look at the models with respect to the F-measure (Equation (2.5)) for $k = 1, \ldots, 10$ (see Figure 2.4) reveals that the hybrid RS outperforms all other RSs. This result indicates that there exists a certain correlation between user/item metadata and the trips booked. However, the baseline model recommending customers' most recently booked trip outperforms even the hybrid RS.

Another insight that the plot provides is that choosing small list lengths should be favored over choosing long ones (if the application allows), as the F-measures are higher for small list lengths.

Figure 2.4: F-measures of unary ratings-based RSs at $k = 1, \ldots, 10$



This plot shows that unary ratings-based recommender systems (RSs) that were trained using a hybrid RS outperform (in terms of the F-measure as stated in Equation (2.5)) those RSs that were trained without additional data on users and items. The plot also shows that the popularity-based model performs worst. The second baseline, recommending customers' most recently booked trip, however, outperforms all the RSs. In addition, the plot suggests choosing a small list length (if the application of the RS allows), as it provides higher F-measures.

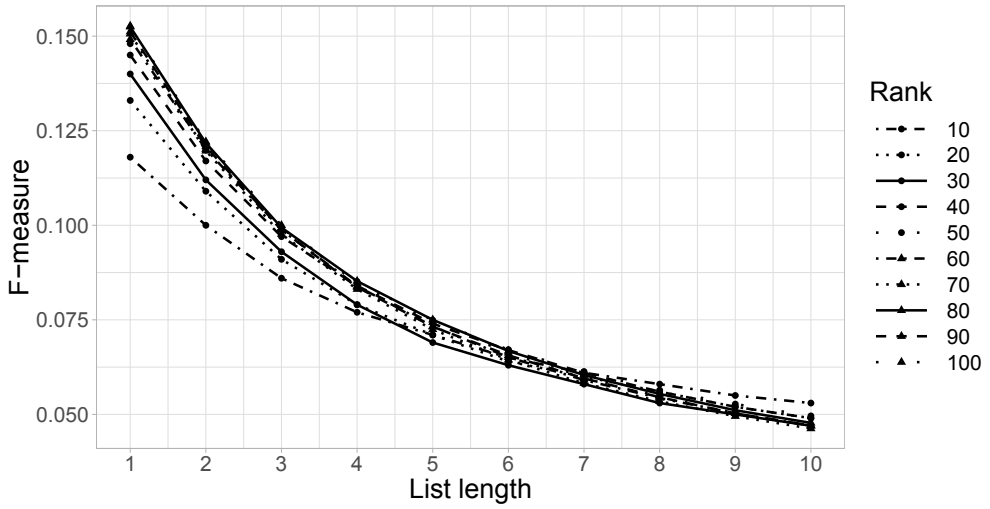
2.4.3 Prediction of Ratings

Is there a way to improve these numbers even further? As already stated, no customer ratings are available for the trips those customers booked. But what can be analyzed instead is how often customers took a trip in the same category (region). This information could serve as an estimate of customers' ratings. So instead of training the models on the unary ratings matrix, we can train them on the (normalized) predicted ratings matrix and see how this affects performance.

Repeating the analyses from above, we again first identify the optimal decomposition rank for matrix factorization. The results are shown in Figure 2.5. This time around, a decomposition rank of $d = 80$ turns out to be optimal.

The optimal combination of the three models' scores is again determined using the validation set approach. The resulting hybrid model again outperforms the other RSs in terms of the

Figure 2.5: F-measure values of predicted ratings-based RSs trained using matrix factorization at $k = 1, \ldots, 10$ for different decomposition ranks d



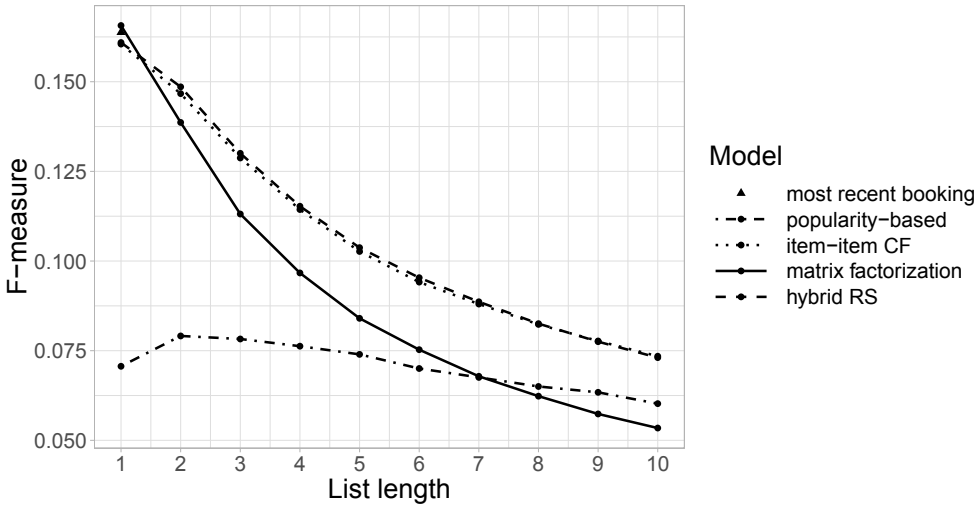
This plot shows the validation performances of predicted ratings-based recommender systems that were trained using matrix factorization. According to our analyses, a decomposition rank of $d = 80$ performs best (in terms of the F-measure as stated in Equation (2.5)) over all list lengths.

F-measure (see Figure 2.6). Hence, RSs that are trained on (predicted) ratings also benefit from additional metadata. Unlike previously though, the baseline model recommending customers' most recently booked trip is no longer the best-performing model. At a list length of $k = 1$, RSs that are trained using the matrix factorization approach outperform the abovementioned baseline model. Once $k > 1$, the matrix factorization model's performance drops and the hybrid RS becomes the best-performing model. Again, as in the unary-ratings case, the plot suggests choosing small list lengths (if the application of the RS allows), as this provides higher F-measures.

2.5 Discussion

As we have seen in the analyses, based on data from D-rt Groep B.V., RSs are able to outperform strategies such as recommending the most recently booked trip or the most popular trips. Beating the former strategy—one of our two baselines—proved, however, to

Figure 2.6: F-measures of predicted ratings-based RSs at $k = 1, \ldots, 10$



This plot shows that predicted ratings-based recommender systems (RSs) that were trained using matrix factorization perform best (in terms of the F-measure as stated in Equation (2.5)) when a list length of $k = 1$ is chosen. For $k > 1$, the hybrid RS takes the lead. In addition, the plot suggests choosing a small list length (if the application of the RS allows), as this results in higher F-measures.

be very difficult. This can, though, be due to the fact that this approach has been used often in the past and therefore that its legacy is reflected in the data, specifically when testing the models' performance on that data. Either way, this approach brings with it the drawback that no new regions will ever be recommended and—hence—it does not provide a big added value for travel agents. In addition, we have seen that the often deployed popularity-based approach clearly performs worse than the RSs presented here.

With the framework provided in this paper, travel agencies should be able to set up an RS based on their own data. If a more sophisticated model is desired, one can train that model on predicted ratings rather than unary ones. In this paper, we look at how many times customers have booked trips in the same category (region) to obtain a proxy for these customers' ratings. From the analyses performed in this paper, we see that RSs that are trained on predicted ratings outperform those trained on unary ratings. Essentially, when training on approximated ratings we simply use the information that is anyway available to us but prepare it in order to use it in a more effective way, which is a significant benefit.

Also, we have seen that including customer and item metadata in the training step results in a model that slightly outperforms classical RSs that are based on ratings alone. Depending on what data is included, this effect can become stronger. Most of the user metadata we have used is aggregated data (data on the place of residence). This implies that there exist correlations between aggregated user metadata and the trips that are booked; this could mean, for example, that residents of a more urban area tend to travel to destinations other than those visited by residents of more rural areas. Including more data on the individual level would, however, certainly be helpful.

As we have seen from Figures 2.4 and 2.6, the list length k should not be chosen to be too large (if the application of the RS allows). It may be that this result (and also the optimal decomposition rank for matrix factorization) differs for data from other travel agencies. The instructions provided in this paper, however, enable these other travel agencies—and even companies active in other industries—to set up and tune their own RSs.

2.6 Conclusion

The goal of this paper was to show how brick-and-mortar travel agencies can improve their businesses in order to not lose further market share to online travel agencies. Classical travel agencies have collected a lot of data but the great majority of these agencies still do not know how to make use of it. The paper proposes the use of recommender systems (RSs), and compares their performance to that of two often deployed approaches: recommending customers' most recently booked trip, and recommending trending destinations (popularity-based approach). To the best of our knowledge, this is the first study that shows how to set up RSs for brick-and-mortar travel agencies.

The analyses show that even the most basic RSs (trained using the neighborhood approach or matrix factorization) are able to outperform the popularity-based model. If, in addition, metadata on users and items is added (even if it is on an aggregated level only), the performance of the RSs can be further improved. They are not, however, able to outperform our

second baseline—recommending customers' most recently booked trip.

As is probably the case for most travel agencies, the data does not contain any explicit ratings from customers with regard to the trips they booked, only the fact that they have booked them—a ratings category that is referred to as unary ratings. The paper shows that these ratings can, however, be predicted, by counting the number of trips that customers booked per category (region). Comparing the RSs that are trained on unary ratings with those trained on predicted ratings shows that predicting ratings is of great benefit: the resulting model was able to outperform the baseline model that we could not beat using unary ratings alone. Thus, to any travel agency currently using either or both of our two baseline approaches to propose trips to their customers we strongly recommend replacing these approaches with RSs.

This paper provides a framework for travel agencies—and also for companies from other industries—that explains how to set up an RS. It would be interesting to see if the findings of this paper can be generalized for travel agencies. In addition, it would be interesting to implement the proposed RSs in travel agencies and measure their true impact, for example by measuring the conversion rate of a campaign such as the one presented in the introduction to this paper.

Toward Fairer Speech Recognition Models: A Generative Adversarial Networks Approach[3.1]

Vanessa Kummer, University of Zurich
Andreea Hossmann, Swisscom
Alexandros Lazaridis, Swisscom
Karl Schmedders, University of Zurich and IMD
Claudiu Musat, Swisscom

Abstract

Automatic speech recognition (ASR) systems are criticized for decoding the speech of certain groups of people more accurately than that of others. Our aim is to analyze how "fair" ASR systems are. A given algorithm is said to be fair if its results are independent of certain variables we consider to be sensitive—for example gender or ethnicity. We show that when, as is often the case, mel-frequency cepstral coefficients (MFCCs) are used to extract features from speech and then to train speech recognition models, the resulting models suffer—due to the nature of how MFCCs are set up—from disparate treatment. In addition, we analyze how imbalances in the amount of speech per group in the training corpus affect the decoding performance per group. This is especially concerning because imbalances are practically impossible to entirely remove. We show that models perform worse in decoding the speech of groups that are underrepresented in the training corpus. This implies that models suffer from disparate impact. To reduce the "unfairness" resulting from the use of MFCCs and from an imbalanced training corpus, we suggest a wrapper that transforms MFCCs extracted from the speech of speakers with certain underlying accents into that of speakers whose accent was predominant in the speech used for training the ASR system, so that the MFCCs are indistinguishable from one another. To achieve this, we propose the use generative adversarial networks.

[3.1]We thank Wissem Allouchi, Marlon Azinovic, Luca Gaegauf, Emma Glaude, and Isaac Leimgruber for their thoughtful support. We are also grateful to everyone at Swisscom and the University of Zurich who provided us with their valuable feedback on our work.

3.1 Introduction

We are in an age where many things have become or are becoming automated. Machine learning (ML) algorithms provide us with movie recommendations, suggest which products to buy, and decide whether we should receive a loan or get that job. They are used in criminal justice, in a range of medical applications, and in autonomous vehicles. This development obviously has many benefits for us humans: unlike people, for example, machines do not suffer from tiredness, and they can consider many more factors than any human can take into account. But since these systems directly affect our lives, they can also harm us if not designed properly. ML systems rely heavily on data for their training—data that humans provide. If the data provided is biased, so will the system be. This fact, together with the widespread use of such systems, has led to "fairness" becoming an important research topic in recent years (Mehrabi et al., 2019).

Anti-discrimination laws in many countries prohibit unfair treatment of people based on sensitive attributes such as gender or race (Civil Rights Act, 1964). These laws typically assess fairness using two notions (Barocas and Selbst, 2016): disparate treatment and disparate impact. A decision involves disparate treatment if it is (even partly) based on sensitive attributes of the subject. Having disparate impact, meanwhile, means that a decision-making process results in outcomes that hurt people who have certain sensitive attributes. The problem is that even if "protected" attributes (so, attributes that are considered sensitive and should thus not be used for training models) are not explicitly used for training a statistical model, other features on which the model is trained may act as proxies of these attributes (for example, neighborhood can be a proxy for race). If such proxies are used to train a model, undesired bias will still result. It is thus sometimes very hard to determine whether a feature that is relevant for a model's fit is too strongly correlated with protected features for it to be used for training that model.

Even more problematic, in certain tasks such as in speech recognition the coefficients used for training are sometimes partially based on protected attributes. Speech recognition systems

are often trained on mel-frequency cepstral coefficients (MFCCs), speech signal representations developed by Bridle and Brown (1974) and Mermelstein (1976). Depending on an individual's vocal tract and many other features, waves oscillate in unique patterns. Since MFCCs aim to separate speech into its source and system components, by their very nature they represent speech in terms of the attributes of speakers. This implies disparate treatment. We will show this by proving that there is a correlation between the similarity of speakers' MFCCs and that of their protected attributes.

Furthermore, we will show that disparate impact too occurs in speech recognition models if the training set is not balanced with respect to all groups either trained on or tested (in our case, accents). We do so by training speech recognition models on subsets of the LibriSpeech (Panayotov et al., 2015) corpus and decoding VoxForge speech. In the LibriSpeech subsets we use, speakers have accents that are close to those prevalent in American English (Panayotov et al., 2015), whereas VoxForge speakers have accents ranging from American English through British English to Indian English. Our analyses show that decodings stemming from models trained on LibriSpeech recordings are substantially more accurate for speakers with an American or Canadian English accent than, for example, for speakers with an Indian or Australian English accent. Adding Indian or Australian English speech to the training corpus improves the decoding performance with regard to the respective accent.

Recent efforts to reduce the bias that is induced by training data have focused on building fairer and more diverse data sets (Celis et al., 2016; Yang et al., 2020), but such an approach is often not feasible. In speech, for example, there exist too many different groups (combinations of accent, gender, age group, and other factors) and collecting enough speech per group in order to make the training corpus balanced—hence, closing the gap between the amount of data available for the main training groups and all others—is, due to the costs it implies, not an option. Instead, we suggest including a wrapper in the model that *translates* the feature vector of underrepresented groups such that the model can no longer distinguish which group the feature vector stems from. This is accomplished by adapting the technique

of generative adversarial networks (GANs).

The remainder of the paper is organized as follows: In Section 3.2 we give a brief overview of how speech recognition works. Section 3.3 looks at what kinds of fairness measures exist and how they can be adapted to measure the fairness of speech recognition models. In Section 3.4 we introduce our debiasing approach. Section 3.5 covers our analyses and their results. Section 3.6 concludes.

3.2 Speech Recognition

Automatic speech recognition (ASR), the use of machines to translate speech in the form of a recorded audio signal into digitized text (Gruhn et al., 2011), has grown incredibly efficient and accurate in recent years. Today, ASR systems can be found in the home (in the form, for example, of virtual personal assistants or smart TVs), in car navigation systems, in smartphones, and many other appliances and applications (Chen et al., 2017). The problem, though, is that most voice recognition systems are biased, for example toward a particular gender (Howard and Borenstein, 2018). And this despite the fact that in order to be helpful to their users, they must be able to both understand and respond quickly and accurately—with respect to all kinds of people.

Typically, the first step in ASR is to extract a series of features from an audio waveform. Air-pressure waves emanate from the mouth and the nostrils of a speaker, and are the generators of speech (Anne et al., 2015). Depending on the tongue placement in the mouth, lip shape, activation of the vocal chords, and many other factors, waves oscillate in unique patterns. The features that are extracted from audio waveforms represent small frames (typically 25 milliseconds long, shifted by 10 milliseconds each time) of the speech we are interested in.

In a second step, the extracted features are matched to "phones".[3.2] Phones are the units

[3.2]Soltau et al. (2016), for example, train an ASR model on 125,000 hours of speech, resulting in 100,000 words. With such an amount of data, end-to-end models that are based on words can be trained, rather than using the approach we explain here.

of sound that make up words. Each phone corresponds to a symbol, so each word can be given as a series of symbols. To make relevant conclusions about incoming sound signals, the extracted sequence of features is mapped to a sequence of phones using a hidden Markov model, HMM. The distribution of features for each phone is modeled using a Gaussian mixture model (GMM).

In GMMs, feature frames can be mapped either to states of HMMs representing isolated phones as a whole (which is called context-independent (CI) or monophone training) or to states of HMMs representing phones in context. This context is usually represented by a phone's preceding and following phone (triphone HMM), based on the fact that a phone is pronounced differently depending on what phone precedes it and follows it. The latter mapping is therefore referred to as context-dependent (CD). The choice between CI and CD phone mapping depends on which language we are modeling (how consistently phones are pronounced) and on how many resources one wants to use when building a speech model.

With the help of a pronunciation lexicon, each set of phones is then mapped to the most probable word. Finally, the language model (or "grammar") estimates the probability of a given word sequence. These three steps together form the decoding graph (Senior, 2017).[3.3]

The accuracy of an ASR system is measured by the word error rate (WER). The WER is derived from the Levenshtein distance, working at the word level instead of at the phoneme level. Its equation is given by

$$WER = 100 \times \frac{S + D + I}{N}, \tag{3.1}$$

where S is the number of substitutions, D the number of deletions, and I the number of insertions between the output of the ASR system (decoding) and the reference transcription (having N number of words), after the decoded text has been aligned with the reference in an error-minimizing way.

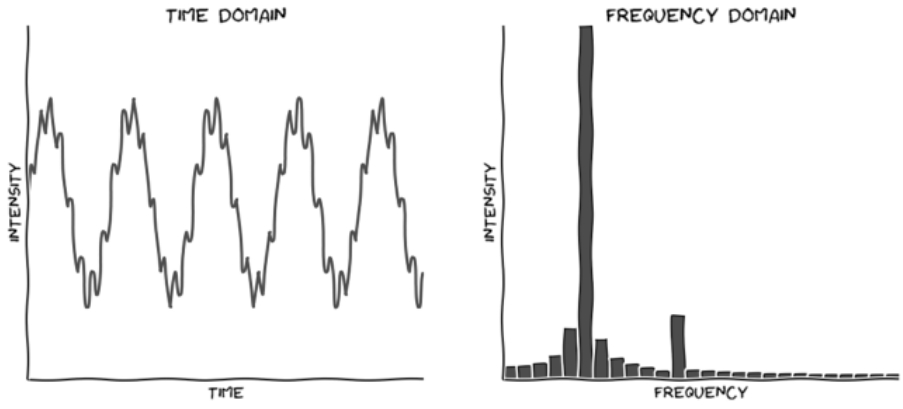
The software we use to create our ASR systems is called Kaldi. Kaldi has been developed

[3.3]For further details on ASR, consult, for example, Gruhn et al. (2011).

by Povey et al. (2011) and is an open-source speech recognition toolkit written in C++, freely available under the Apache License v2.0. The software is intended for use by ASR researchers.

Kaldi uses the previously mentioned mel-frequency cepstral coefficients (MFCCs) for feature extraction (Povey et al., 2011). The first step in creating MFCCs is to identify the frequency of a signal (audio wave). Technically, one could look at the amplitude value of the signal at every time step. But since the sampling rate would have to be very high in order to reduce the loss of information, a lot of computational space would be required. A better approach is to represent signals in the frequency domain by applying a discrete Fourier transform (also known as a fast Fourier transform). In general, Fourier transforms allow us to identify the frequency of a signal, as any periodic signal shows up as a sharp peak in the corresponding frequency spectrum (see Figure 3.1).

Figure 3.1: Fourier transform on time signal



The figure demonstrates that the frequency of a signal (audio wave) can be identified by applying a Fourier transform. The signal's frequency shows up as a sharp peak in the corresponding frequency spectrum. This step is essential to the derivation of MFCCs from audio signals. Source: Nair (2018).

Next, we have to convert the frequency to mel scale (often referred to as mel filtering). This helps to match the frequency more closely to what the human ear can hear (humans are better at identifying small changes in speech at lower frequencies—the perceived distance

between a 900 Hz and a 1 kHz signal is smaller than the perceived distance between a 300 Hz and a 400 Hz signal). A frequency measured in Hertz, f, can be converted to the mel scale using Equation (3.2), an increasing and concave function.

$$Mel(f) = 2595 \log_{10} \left(1 + \frac{f}{700}\right). \tag{3.2}$$

Because the resulting features are of too high a dimension, a discrete cosine transform of the mel frequency is taken, leading to the desired cepstral coefficients (MFCCs). Typically, GMM speech recognition systems are trained on the first 13 MFCCs (Nair, 2018; Senior, 2017).

3.3 Fairness Measures

Speech recognition systems are often criticized for decoding the speech of certain groups of people more accurately than that of others. Tatman (2017), for example, evaluates the accuracy of YouTube's automatically generated captions and finds that the system produces reliably less accurate decodings for speakers from Scotland (compared to speakers from the United States or New Zealand). The author also shows that the system performs worse in decoding female speech compared to male speech (Tatman, 2017). More recently, Koenecke et al. (2020) published a paper demonstrating large racial disparities in the performance of five popular commercial ASR systems.

When one thinks about bias in machine learning, a question that arises almost immediately is: "Isn't discriminating between data points the very point of machine learning?" And while this is, of course, true, bias is essentially induced in models through the human bias existing in the data set. Skewed or tainted samples, but also sample size disparity, can therefore already cause a model to be biased (Barocas and Selbst, 2016). Hence, just because something has statistical relevance that does not mean it should also be used for decision-making.

Research on fairness in machine learning has dramatically increased over the last few years—

in academia as well as in industry (Du et al., 2019). Crawford argued in her talk at NIPS 2017 (Crawford, 2017) that so far most attention has been paid to allocative harm (for example, decisions about who gets a loan and who does not) rather than to representational harm (occurring, for example, in image or speech recognition). For Crawford, the reason for this is that allocative harm has an immediate effect and is easily quantifiable, whereas representational bias is much more difficult to formalize. In the years that followed the conference this changed. Many research papers published in 2018 or 2019 cover the mitigation of bias in image—especially face—recognition (these include Amini et al. (2019) and Sattigeri et al. (2019)). Less attention was, however, paid to mitigating bias in speech recognition models.

To formalize fairness in ASR systems, we need to reformulate traditional fairness measures. Typically, a model's fairness is evaluated using the notions of disparate treatment and disparate impact (Barocas and Selbst, 2016). A decision-making process suffers from disparate treatment if its decisions are (partly) based on a subject's sensitive attributes; it has disparate impact if its outcomes disproportionately hurt (or benefit) people with certain sensitive attribute values (such as gender or race). One of the basic criteria for measuring fairness is independence (also sometimes referred to as demographic parity). A classifier that makes a binary class prediction $\hat{y} \in \{0, 1\}$ is independent of a binary sensitive attribute $z \in \{0, 1\}$—for example, gender—if

$$P\{\hat{y} = 1|z = 1\} = P\{\hat{y} = 1|z = 0\}, \tag{3.3}$$

for $z = 1$ being the positive (or negative) outcome.

Fairness comes at the cost of reduced model accuracy though. Independence is therefore often relaxed by a certain degree. Zafar et al. (2017) define the so-called p%-rule. A classifier satisfies the p%-rule if the following inequality holds:

$$\min\left(\frac{P\{\hat{y} = 1|z = 1\}}{P\{\hat{y} = 1|z = 0\}}, \frac{P\{\hat{y} = 1|z = 0\}}{P\{\hat{y} = 1|z = 1\}} \geq \frac{p}{100}\right). \tag{3.4}$$

The usual choice for p is 80[3.4], which is why one often sees reference to the 80 percent or four-fifths rule when reading about fairness. This implies that the ratio between the probability of a positive outcome given the sensitive attribute being true and the same probability given the sensitive attribute being false is not allowed to be less than 80 percent. A classifier that would satisfy the 100 percent rule would hence be completely fair, whereas a classifier that would satisfy the 0 percent rule alone would be completely unfair.

An alternative criterion is *individual* fairness. The independence criterion, which we have just examined, is group based, while individual fairness, as its name suggests, is individual based. It was first proposed by Dwork et al. (2012). The emphasis in individual fairness is that similar individuals should be treated similarly. The criterion was introduced because in certain cases, even though group fairness is satisfied, from the point of view of an individual the outcome is blatantly unfair. The individual fairness criterion sounds extremely obvious, but it comes with a big difficulty: how to determine if people are similar. Are two people, one having an American accent and being male and the other having a Canadian accent and being male, more similar to one another than are two people with an American accent, one being male and the other female? Or is a Scottish accent more similar to an American accent than it is to an Australian accent? These questions make it very difficult to quantify people's similarity to one another. For this reason we focus on group fairness in the following.

Since we have multiple sensitive attributes (such as gender or accent), we can create attribute groups X. The reference group is that with the lowest WER. All other groups' performances are then compared to that of the reference group. In the optimal case (perfect speech recognition) the WER would equal zero. In Equations (3.3) and (3.4), $P\{\hat{y} = 1|z\}$ is linked to a more preferable outcome the higher the percentage is (high probability of getting a

[3.4]In 1978, four US government agencies (the Equal Employment Opportunity Commission, Department of Labor, Department of Justice, and Civil Service Commission) adopted a set of guidelines (*Uniform Guidelines*) for employee selection procedures. The guidelines provided information on what constitutes a discriminatory test in the context of recruitment and broader personnel matters. Disparate impact in personnel processes was determined using the four-fifths or 80 percent rule, which is why this number became the standard. (EEOC, 1979)

job, for example). Hence, in order to adapt the measures for speech recognition models, $P\{\hat{y} = 1|z\}$ can be translated to $1 - WER_{X=x}$. This results in the following adaptation of Equation (3.3):

$$1 - WER_{X=x} = 1 - WER_{X=ref}, \tag{3.5}$$

for all groups $x \in X$. Equivalently, Equation (3.4) can be translated for speech recognition model purposes as follows:

$$\min\left(\frac{1 - WER_{X=x}}{1 - WER_{X=ref}}, \frac{1 - WER_{X=ref}}{1 - WER_{X=x}} \geq \frac{p}{100}\right). \tag{3.6}$$

In general, a model's fairness can be improved by modifying the training data (*pre-processing*), the learning algorithm (*in-processing*), or the predictions (*post-processing*). The choice of which is based on the ability to intervene at different points in a machine learning pipeline (Bellamy et al., 2019). Examples for pre-processing algorithms include the re-weighting of training examples per group and under-/oversampling (Kamiran and Calders, 2012). In-processing algorithms include adversarial debiasing, which learns a classifier to maximize prediction accuracy and simultaneously reduce an adversary's ability to determine the protected attribute from the predictions (Zhang et al., 2018). If the algorithm can only treat the learned model as a black box and has no ability to modify the training data or learning algorithm, then only post-processing can be used (Bellamy et al., 2019). An example for post-processing is finding the optimal classification threshold by looking at two groups' receiver operating characteristic (ROC) curves. Equalized odds requires that the groups' true positive rates be equal, and that the same be true of their false positive rates. Hence, the classification threshold should be set to the value where the two ROC curves intersect (Hardt et al., 2016).

Since a lot of progress has already been made in training ASR models, we focus on how the fairness of already trained models can be increased. We therefore introduce a wrapper: an additional step the MFCCs have to pass, in which they are modified, right before they are

decoded by the trained ASR model. Thus, we are not carrying out classical in-processing or post-processing. Instead, our approach lies somewhere in between.

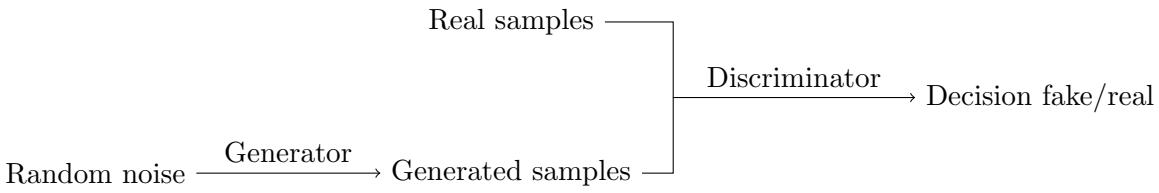
3.4 Debiasing Approach

The most straightforward approach to increasing fairness across groups of people would be to make the training set balanced. This would imply increasing or decreasing the available amount of speech per accent until the corpus is balanced. Undersampling majority classes (deleting examples from the majority class) often lowers the accuracy of the model however. In addition, the more imbalanced the data set, the more samples will be discarded when undersampling, therefore throwing away potentially useful information. Increasing data for minority classes is often not possible unless we oversample, which would imply duplications of samples from speakers from minority classes. Our own preliminary analyses of training ASR systems on an oversampled corpus have revealed a decrease in model performance, probably due to overfitting.

The approach we present here allows us to use the original features extracted from speech (MFCCs), but instead of changing the amount of training data per group we build a wrapper (an additional step before decoding) that acoustically brings the speech of speakers with an accent other than that mainly used for training (in our case, American English) closer to speech that exhibits that predominant accent.

In 2014, Goodfellow et al. published a paper on generative adversarial networks (GANs). It introduces GANs as a system of two neural networks—a generative model and an adversarial classifier—which are competing with each other. The generative model focuses on producing samples that are indistinguishable from real data, while the adversarial model tries to identify if samples came from the generative model or from the real data (illustrated in Figure 3.2). Both networks are trained simultaneously such that the first improves at producing realistic samples, while the second becomes better at spotting the "fake" samples (Goodfellow et al., 2014).

Figure 3.2: Schematic representation of a GAN

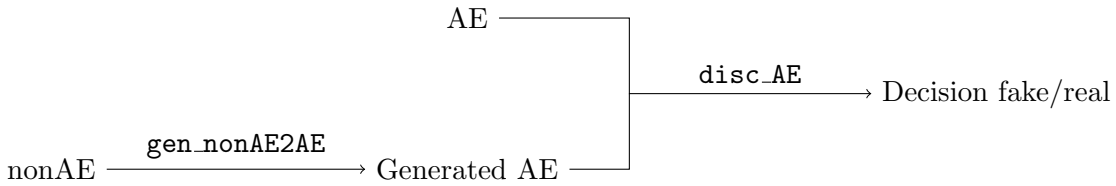


This figure illustrates how GANs work: The generator is fed with random noise and tries to create an output that is indistinguishable from a real sample. The discriminator is fed with real as well as generated samples and tries to identify which samples are real/fake.

We adapt this technique by feeding the generator with MFCCs extracted from recordings of speakers with a certain accent other than American English (in the following referred to as non-AE), teaching it to produce MFCCs that are indistinguishable from those of recordings of American English (AE) speakers. For each accent other than AE, a separate model can be trained (we will, though, only analyze one accent). The adversarial classifier tries to identify if the coefficients came from speakers with an AE accent or not (the latter meaning that they were produced from the generative model). Training the generator requires that we assess its performance on the output of the discriminator. Both networks therefore have to be added to a combined model: an adversarial model. Our adversarial network uses MFCCs from the recorded utterances of non-AE speakers as well as those of AE speakers as its input, and classifies—after MFCCs of non-AE speakers have been modified by the generator—whether the MFCCs stem from AE speakers or whether they stem from non-AE speakers and have been modified by the generator (see Figure 3.3). The generator performs well if the adversarial model outputs "AE speaker" in response to all inputs—true MFCCs from AE speakers but also generated MFCCs. This, in turn, means that the discriminator has failed in its task. If we used normal back propagation for training the adversarial model, we would force the discriminator to start classifying all MFCCs as AE speakers' MFCCs. To prevent this, we must freeze the part of the model that belongs to the discriminator. This means that we need to prevent the discriminator from updating in the adversarial model,

but have it remain trainable in the discriminator model.

Figure 3.3: Initial GAN implementation



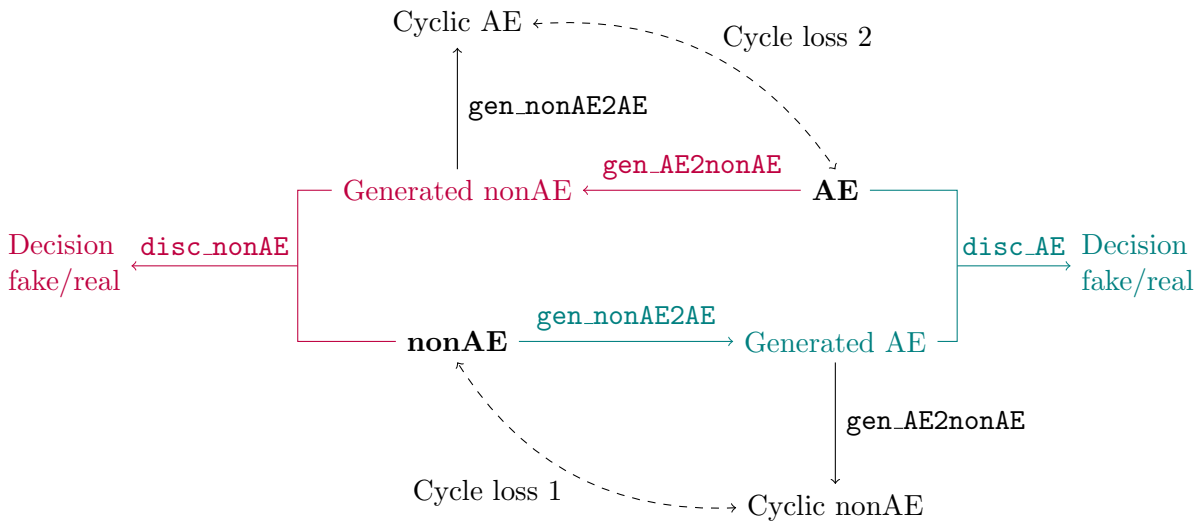
The architecture presented in this Figure is that of the initial GAN we set up. The generator, `gen_nonAE2AE`, takes as input MFCCs of non-AE speakers' speech and generates MFCCs that resemble those of AE speakers. The discriminator, `disc_AE`, is fed with these generated AE MFCCs and with real ones, and tries to predict which MFCCs are generated and which are real.

This method of training the adversarial network has a problem though. We are not just interested in generating speech that resembles AE speech—we have to ensure that the content that is embedded in the MFCCs does not change. At some point in the training process however, the generator always starts to map all input MFCCs to the same output, which then, of course, no longer contains the same speech content as the input MFCCs. This phenomenon, referred to as *mode collapse*, is a big challenge when training GANs (Goodfellow, 2016). To solve this problem, Zhu et al. (2017) introduce the constraint of "cycle consistency".

Figure 3.4 gives an overview of our CycleGAN architecture. Different from the classical GAN, *two* adversarial networks—each consisting of a generator and a discriminator—are trained. Hence, we not only train the network to perform well for the direction non-AE to AE-resembling MFCCs, but also for the opposite direction. These networks are, however, interrelated: one network's trained generator is used by the other network to generate cyclic MFCCs. These can then be compared to the original input. Besides the discriminator and adversary losses, we therefore also have two cycle losses (defined as mean absolute errors) that have to be minimized during the training stage. Keeping the cycle loss low ensures the necessary cycle consistency and hence that there is some meaningful relation between input MFCCs and generated MFCCs.

Both generator and discriminator are set up as convolutional neural networks (CNNs). CNNs

Figure 3.4: CycleGAN architecture



This figure illustrates how our CycleGAN is set up: The network consists of two adversarial networks (shown in purple and turquoise, where the turquoise-colored network is the same as the one shown in Figure 3.3), each having a generator (denoted by `gen`) and a discriminator (denoted by `disc`). To train the discriminators and the generators, not only do the discriminator and adversarial losses have to be minimized, so-called cycle losses must too. The idea is that if we generate samples from an input and then back again, we should get samples close to the original input samples. This ensures that the speech content of the input MFCCs does not change.

are fully connected networks—that is, each neuron in one layer is connected to all neurons in the next layer. To have an initial architecture and training setup that we can later adapt, we look at the well-known example of generating fake handwritten digits (Bulten, 2017). As in that setup, we define our network using Keras. Also, we decide to initially keep the leaky version of a rectified linear unit (`LeakyReLU`) as an activation function between the convolution layers. ReLU is the most widely used activation function in neural networks today. One of the great advantages ReLU has over other activation functions (like tanh or sigmoid) is that it does not activate all neurons at the same time. This makes it very computationally efficient as few neurons are activated at any given time.

Moving beyond that initial, well-known example, we define our generator as an encoder–decoder network: The first few layers (the encoder) extract features in an unsupervised manner from the MFCCs. In other words, the encoder learns a lower-dimensional repre-

sentation of the original data. The next few layers (the decoder) combine these extracted features to produce the desired output. The structure of the decoder network is usually the same as that of the decoder, but set up in the opposite direction.

An additional parameter we have to tune is the weight distribution of the adversarial and the cycle losses. Because it is crucial for us to have a high cycle consistency (to maintain the content of the speech inherent in the MFCCs we are translating), the cycle loss weight has to be chosen to be higher than that of the adversarial loss. This, however, increases the adversarial loss because we do not allow the network to make too significant changes to the input MFCCs.

3.5 Experimental Setup and Results

As already mentioned, we use the open-source speech recognition toolkit Kaldi for training our ASR systems. In multiple steps, the software builds several speech recognition models of increasing complexity. We work with context-dependent (CD) HMM/GMM hybrids. In addition, speaker adaptive training (SAT) is added. Models trained with SAT become independent of their training speakers and therefore generalize better to their testing speakers (Miao et al., 2015).

3.5.1 Speech Corpus

A classical data set for training ASR systems is the freely available LibriSpeech corpus from Panayotov et al. (2015). The data is derived from audiobooks and contains 1,000 hours of speech, along with the corresponding transcripts and information on the speakers' genders. Because the size of the corpus may prove computationally troublesome for some users, the authors split the training data into three subsets with approximately 100, 360, and 500 hours of speech, respectively. Recordings in the first two sets are, on average, of higher quality, and the accents are closer to American English. Also, the authors have ensured gender balance at the speaker level and in terms of the amount of data available for each gender (Panayotov

et al., 2015).

A speech corpus that not only contains information on speakers' gender but also on their accent and age group is the VoxForge[3.5] corpus. This corpus contains transcribed recordings from various speakers, whereof 4,783 have one of the following accents: American, Australian, British, Canadian, Indian, Irish, New Zealand, Northern Irish, or South African English (an overview of the speakers, after the data has been cleaned, is provided in Table 3.1). Using recordings from these speakers allows us to identify differences in the performance of decoding speech from speakers with different accents—hence, it allows us to identify whether the ASR system is fair.

Table 3.1: VoxForge speakers (spks)

Accent	**Number of spks**	**Minutes of speech**	**Ratio female/male spks**
American	3,020	3,826	244/2,776
Australian	191	249	10/181
British	787	823	66/721
Canadian	343	368	37/306
Indian	237	243	7/230
Irish	33	30	1/32
New Zealand	84	87	3/81
Northern Irish	10	23	0/10
South African	78	66	5/73

This table shows the number of speakers (spks), their genders, as well as the amount of speech per accent-group present in the VoxForge corpus. The majority of speakers has an American English accent, the accent represented least is Northern Irish English. Also, we can see that there are much more male than female speakers in the speech corpus. Data source: `http://www.voxforge.org/`

[3.5] `http://www.voxforge.org/`.

3.5.2 Disparate Impact

To show the effect of sample size disparity on fairness, in a first stage three CD-SAT-HMM/GMMs are trained on a randomly chosen 3.5 hours of LibriSpeech recordings (in the following referred to as `set1`; see Table 3.2). The speech corpus contains clean speech that is close to American English. Once the models are trained, we decode VoxForge speech, which consists of recordings from speakers with a range of English accent types.

Table 3.2: Models and training sets used in our analysis of disparate impact with regard to Indian English

Term	**Explanation**
`Set1`	Random 3.5 hours of American English speech from the LibriSpeech corpus
`Set2`	Random 3.5 hours of Indian English speech from the VoxForge corpus
`Set3`	Random 3.5 hours of American English speech from the LibriSpeech corpus (`set1` excluded)
`Model1`	CD-SAT-HMM/GMM trained on `set1`
`Model2`	CD-SAT-HMM/GMM trained on `set1` + `set2`
`Model3`	CD-SAT-HMM/GMM trained on `set1` + `set3`

This table gives an overview of the training sets used for our analysis of disparate impact, and of the ASR models trained.

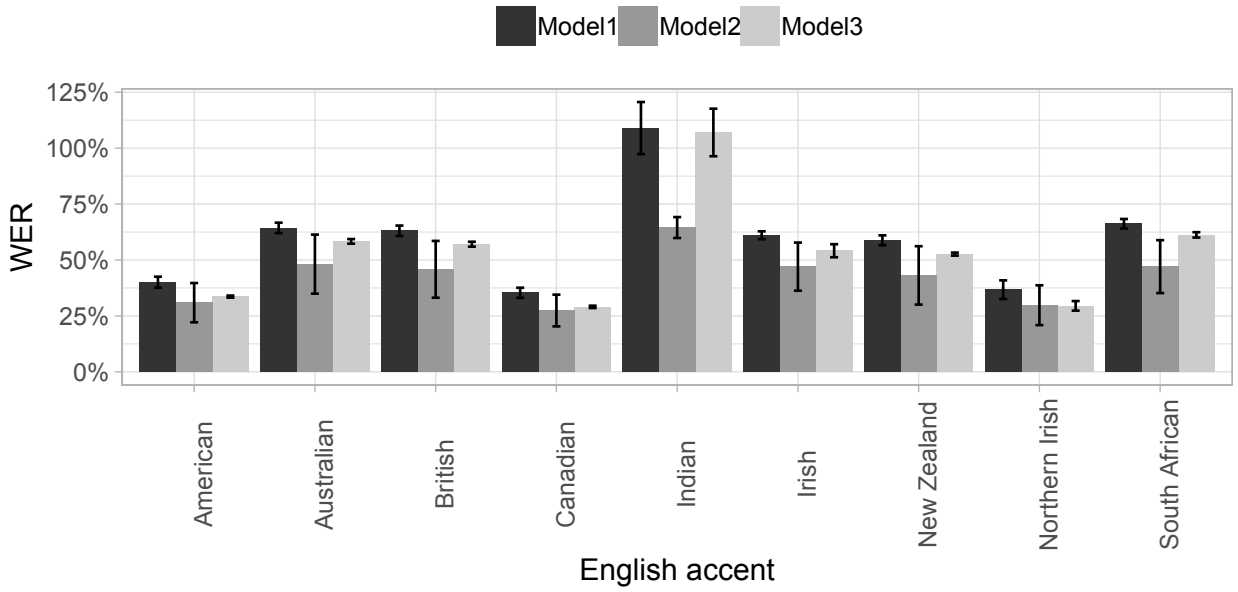
The second set of CD-SAT-HMM/GMMs is trained on the same 3.5 hours of LibriSpeech recordings plus on an additional 3.5 hours of Indian English speech (refer to Table 3.2, `set1` + `set2`). The third and last set of CD-SAT-HMM/GMMs is trained on 7 hours of LibriSpeech speech, 3.5 hours of which are identical to the corpus used for training the first set of models (so, `set1` + `set3`). Again, VoxForge speech is decoded and performances are compared.

The mean performance (WER, as defined in (3.1)) of decoding VoxForge test data with `model1` is 45.83%. Looking at the decoding performance per accent available in the test set (Figure 3.5), we see that the models perform particularly well for American, Canadian,

and Northern Irish English speakers, but very badly for Indian English. This is what we expected though, because there is no Indian English speech present in the training set.
Decoding the same test set with `model2`, the average overall WER falls to 34.66%. As expected, the average WER resulting from decoding Indian English speech has also decreased, from 108.97%—the WER can exceed 100% if the decoding is so poor that the alignment cannot be carried out properly—to 64.51%. It is interesting, though, that the WER of decoding the speech of speakers with any other accent has also decreased. To check whether the decrease in WER is only a consequence of doubling the amount of training data, we train a set of models on seven hours of LibriSpeech data (`model3`). The resulting average test WER is indeed lower (39.51%) than that stemming from `model1` (45.83%), but it is higher than the average WER from `model2` (34.66%). Also, when we look at Figure 3.5, the WERs have decreased for all accents but not to the extent that they do when we decode speech with `model2`.
One reason for this result could be that the LibriSpeech corpus contains a lot of speech per person but not the speech of a lot of people, whereas the VoxForge (and hence, Indian English) corpus contains little speech per person and instead contains speech from many different people. Therefore, when we added 3.5 hours of Indian English recordings (`set2`) to the 3.5 hour subset of LibriSpeech training data (`set1`), the number of speakers increased, approximately from 20 to 200, whereas when we added another 3.5 hours of LibriSpeech recordings (`set3`), the number of speakers increased only to approximately 40. Hence, involving the VoxForge data means that the heterogeneity of speakers is much larger than when we consider LibriSpeech recordings alone. In addition, the LibriSpeech data set, from which we take the random subsets for training, is constructed in such a way that recordings are similar to one another (high recording quality and accents close to American English).

So far we have only looked at performance, but what we are ultimately also interested in is whether the models are fair or not. We therefore check whether they satisfy the four-

Figure 3.5: VoxForge corpus decoding performance

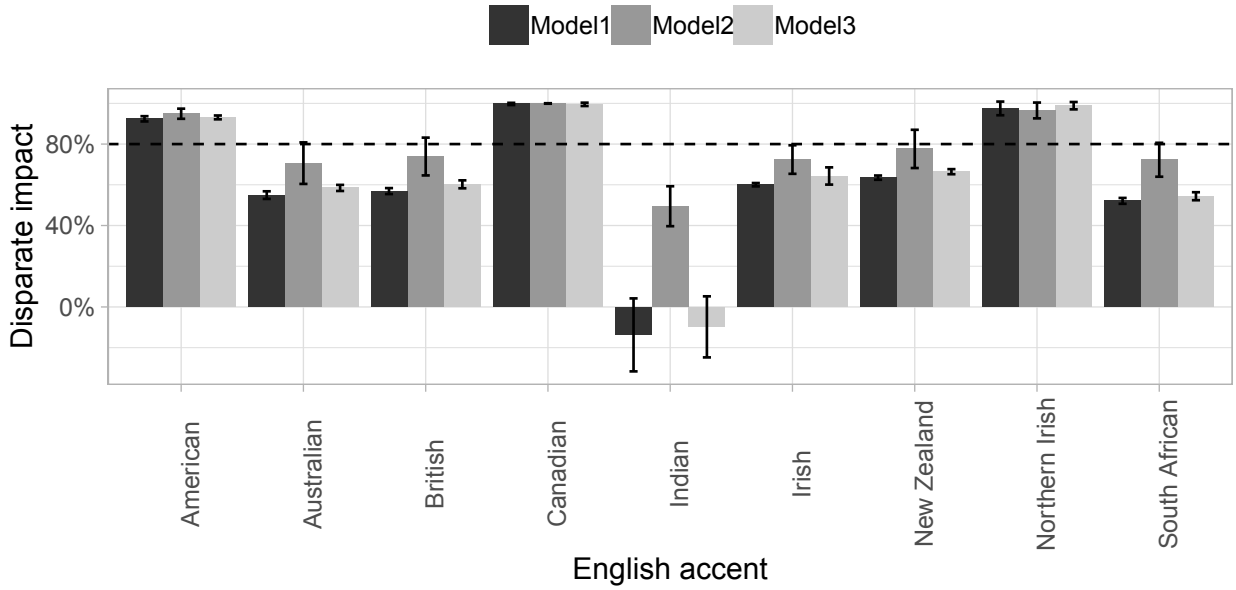


This figure shows that the highest average decoding WERs result from models that are trained on 3.5 hours of LibriSpeech data only (`set1`), followed by models that are trained on 7 hours of LibriSpeech speech (`set1` + `set3`). Adding 3.5 hours of Indian English speakers' speech from the VoxForge corpus (`set1` + `set2`) instead of the additional 3.5 hours of LibriSpeech data improves the decoding performances for all accents, especially for Indian English speakers' recordings.

fifths rule adapted according to Equation (3.6). Figure 3.6 shows disparate impact per accent, Canadian English—the most accurately decoded accent—being the reference. The dashed line is set to 80 percent: the threshold the model has to reach in order to be deemed "fair". We can see that `model1` only satisfies the four-fifths rule for American, Canadian, and Northern Irish English speakers (meaning that it is fair for these accents—and hence not fair overall). Doubling the amount of training speech but sticking to LibriSpeech recordings does not change this result. Only by adding Indian English to the training set do the decoding WERs for speakers with an Australian, British, New Zealand, or South African English accent decrease by so much that their corresponding fairness measure is close to the 80 percent threshold. Considering these fairness measures' standard deviations, the model could be called fair for those accents.

The accent for which the decoding WER decreased most when adding Indian English to the training set is Indian English. Looking at Figure 3.6, we see that fairness has increased significantly for Indian English speakers: the models trained on LibriSpeech recordings do not even fulfill the zero percent rule for Indian English speakers, whereas once we add Indian English to the training set, the 50 percent rule is almost fulfilled. This clearly implies that the bigger the sample size disparity, the more unfair the model. Overall though, because it still fails the four-fifths rule for some accents the model is not fair, and has disparate impact with regard to certain accents.

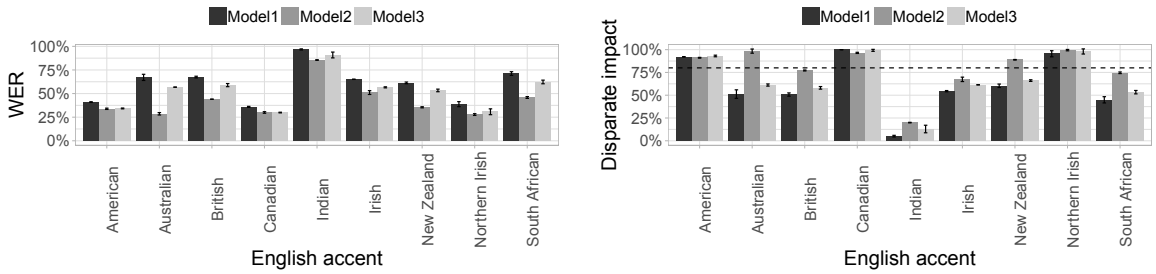
Figure 3.6: Disparate impact on accents



The figure shows how fair the trained ASR models are in decoding the speech of speakers with different accents. A model is deemed fair if it satisfies the four-fifths rule (3.6). We can see that models that are trained on 3.5 hours of LibriSpeech data only (`set1`) are, on average, fair (the 80% threshold is exceeded) only for American, Canadian, and Northern Irish English speakers. This result does not change for models trained on 7 hours of LibriSpeech data (`set1` + `set3`). Models that are trained on 3.5 hours of LibriSpeech data as well as 3.5 hours of Indian English speech—the latter from the VoxForge corpus—(set1 + set2) could be called fair (standard deviations surpass the 80% threshold) for speakers with an Australian, British, New Zealand, or South African English accent. The biggest effect can be seen for Indian English speakers. The analysis shows that too large sample size disparities in the training data can lead to unfairness toward the underrepresented groups.

In order to support the results obtained, we repeat the analysis for Australian English

Figure 3.7: Comparison of decoding performances of the VoxForge speech corpus using LibriSpeech models and an Australian English enriched model



(a) WER (b) Disparate impact

The results shown in these two panels are analogous to those obtained from the previous Indian English analyses: increasing the amount of Australian English speech used for training results in models that are overall fairer than those trained on LibriSpeech data alone. The decrease in WER is largest for the speech of speakers with an Australian English accent.

(results in Figure 3.7). Again, the overall WER decreases when we double the size of the training corpus. Also, we can again see that the overall effect on performance is higher when adding Australian English to the training corpus rather than just adding another 3.5 hours of LibriSpeech data. This effect on decoding performance is largest for the speech of Australian English speakers.

So far we can conclude that if the speech corpus used for training is too imbalanced (groups of speakers are not reasonably equally represented in the training set), the resulting ASR model suffers from disparate impact (unfairness toward the underrepresented groups).

3.5.3 Disparate Treatment

Next, we want to analyze if ASR models trained on MFCCs also suffer from disparate treatment. We do not explicitly include information about speakers' attributes (such as accent or gender) when training the speech model, but we assume that MFCCs are close proxies for those (potentially protected) attributes. This would imply that the resulting ASR model not only suffers from disparate impact but also from disparate treatment, because we include protected attributes during training.

To test our assumption, we check whether the MFCCs extracted from the speech of speakers with similar protected attributes are closer to each other than are those of speakers who are not so similar to each other. Because the VoxForge corpus does not contain recordings of equivalent text for all speakers, we make use of a third speech corpus to perform the analysis of disparate treatment: the CSTR VCTK Corpus of Veaux et al. (2017), which includes recordings of 109 native English speakers with various accents. Of particular interest for us is that every speaker records, among other elements, him- or herself reading *The Rainbow Passage*[3.6] and an elicitation paragraph. In addition, the corpus includes the following metadata about the speakers: age, gender, accent, and region (Veaux et al., 2017).

Because every speaker has a different speed of talking/reading, recordings of the same text, and hence the extracted MFCCs, have different lengths. In order to compare two speakers' MFCCs we therefore use dynamic time warping (DTW).

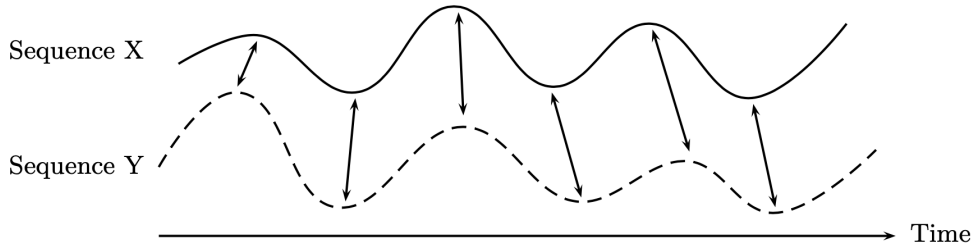
In time series analysis, DTW is used for measuring the similarity between two temporal sequences that may vary in speed. Thanks to its versatility, DTW is also used in many other scientific fields, including biology, economics, and robotics. Almost all DTW methods are based on the original DTW algorithm by Sakoe and Chiba (1978): two sequences are aligned in such a way that—satisfying monotonicity, boundary, and continuity constraints—the sum of the absolute differences between two sequences' values, for each matched pair of indices, is minimal (as in Figure 3.8).

The monotonicity constraint ensures that the path represents a monotone increasing function of time. The boundary constraint enforces that the first (last) index from one sequence must be matched with the first (last) index from the other sequence, and vice versa. The warping path hence begins (ends) with the origin (terminal) points of both sequences. Finally, the continuity constraint ensures that every index from one sequence must be matched with one or more indices from the other sequence, and vice versa (Sakoe and Chiba, 1978).

We use this technique to find the optimal alignment between two speakers' MFCCs from

[3.6] `http://www.dialectsarchive.com/wp-content/uploads/2015/11/The-Rainbow-Passage.pdf`.

Figure 3.8: Dynamic time warping



DTW is used for measuring the similarity between two temporal sequences that may vary in speed. We use this concept to measure the similarity between two speakers' MFCCs from recordings of them reading the same text. This allows us to analyze whether two speakers that are relatively similar to one another in terms of their attributes (such as accents, age, or gender) also have MFCCs that are relatively similar one another. That would imply that MFCCs suffer from disparate treatment. Graphic source: Müller (2007)

an utterance, and, based on the found alignment, calculate the distance between the two MFCCs. We then compare whether the average distance between the MFCCs of speakers with the same accent differs significantly from that of speakers with different accents. The results can be found in Figure 3.9.

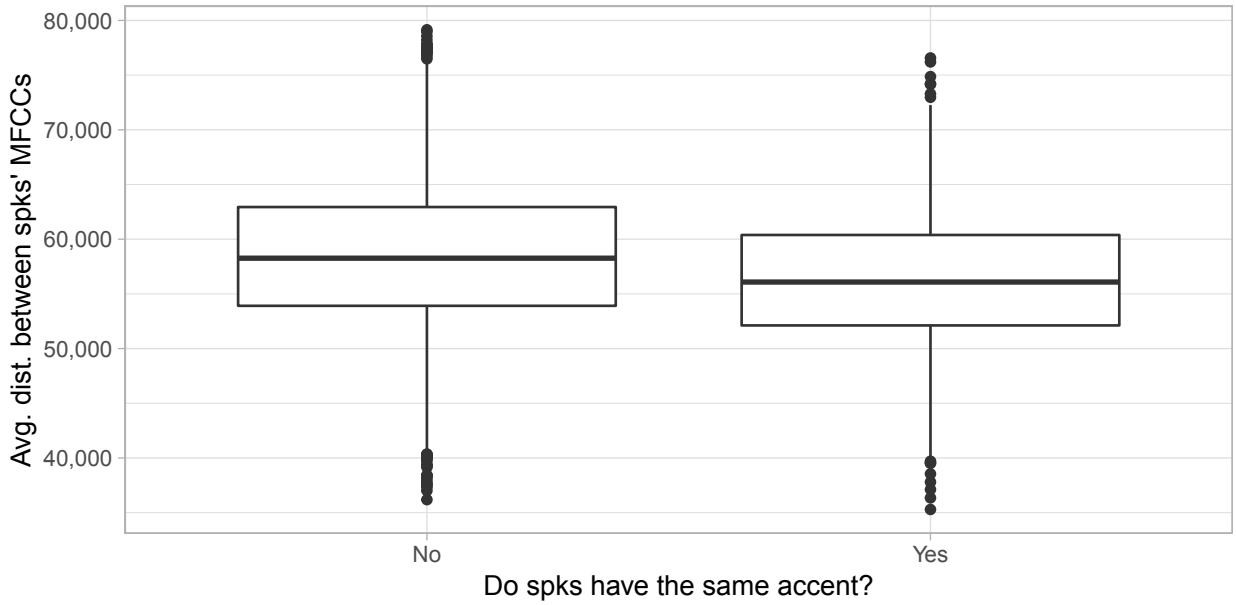
We perform a two-sample Kolmogorov–Smirnov test to check whether the average distance between the two speaker groups' MFCCs has the same distribution. This test results in a p-value of 3.488e-13, suggesting that we can reject H_0—that the two average distances stem from the same distribution. With a two-sided t-test we analyze whether the means are identical. Also in this test the H_0 can be rejected, with a p-value of $< 2.2e - 16$.

Hence, it is safe to say that MFCCs are indeed correlated to, potentially sensitive, attributes. This implies that even though we do not include sensitive attributes explicitly in the training step, by using MFCCs the resulting model for ASR will include this information. In what follows, we will show the results of debiasing our ASR using the approach of GANs.

3.5.4 Generative Adversarial Network

While trying to train our CycleGAN, we faced huge instability and convergence problems. Brief research revealed several papers in which authors describe their difficulties in training

Figure 3.9: Distance between speakers' MFCCs



The boxplot shows that the average MFCC distance of speakers with the same accent is smaller than that of speakers with different accents. Together with the other analyses performed, this allows us to say that MFCCs are indeed correlated to speakers' attributes, which means that MFCCs suffer from disparate treatment. Data source: Veaux et al. (2017)

GANs. One big challenge is that there exist a lot of tuning possibilities, starting from the architectures of the two networks themselves (including normalization, filter size, and number of layers) and ranging to the loss functions, learning rates, their decay, and many other factors. Finding a Nash equilibrium between two competing neural networks whose cost functions are non-convex and parameters are continuous and whose parameter spaces are extremely high-dimensional is a very difficult task and therefore often results in non-convergence (Goodfellow, 2016; Salimans et al., 2016). Another problem specific to our example is the size of the input data: our MFCC matrices are of dimension $x \times 13$ (13 being the number of coefficients normally extracted from speech), where x often reaches around 10,000. This not only leads to a huge increase in required computational time, making the tuning process very slow, it also causes problems due to the limits of memory size. And yet another problem is that there are no good objective metrics for evaluating whether a

GAN is performing well during training. Instead, a common approach is to visually inspect generated examples and use subjective evaluation (Salimans et al., 2016). This might work well when working with images, but with speech the generated MFCCs have to be fed back to the ASR model after being modified, and decoded by matching the features to phones, phones to the most probable words, and words to the most probable word sequence. Direct model comparison is therefore not possible.

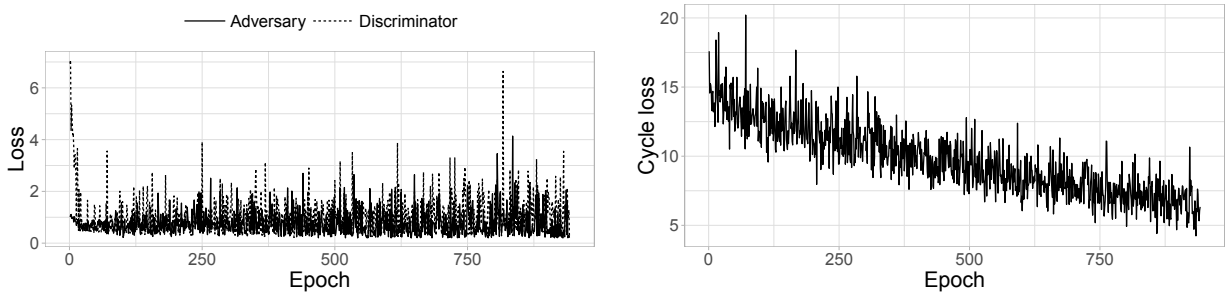
Trials involving changing the two networks to LSTM ("long short-term memory", a recurrent neural network architecture used in the field of deep learning) rather than CNN did not bring any improvement. And neither did changing the loss function improve our results—a fact foreseen by Qin et al. (2018), who show that their GAN is able to learn in every loss setting and conclude that the choice of loss function does not really matter. Therefore, we stick to binary-crossentropy loss, which is defined as

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)], \tag{3.7}$$

y representing the vector of true labels and $\hat{y}$ their predictions.

Something that did improve our model was introducing label smoothing. Classically, if we face a binary classification problem, we label one of the outputs with 1 and the other with 0. Neural networks, however, have the habit of becoming overconfident in their predictions during training. This can reduce their ability to generalize and thus perform as well on new, unseen future data. Label smoothing forces the neural network to be less confident in its answers by training it to move toward a smoothed target. This has an effect similar to that of regularization, hence reducing overfitting. We choose a label smoothing value of 0.1. Each real MFCC (stemming from AE speakers) was therefore labeled with 0.9 rather than 1; every modified MFCC resulting as output from the generator was labeled with 0.1 rather than 0. The effect of label smoothing was a small increase in the discriminator loss, but at the same time it led to a big decrease in the adversarial loss. In addition, training became more stable.

Figure 3.10: Evolvement of losses during GAN training



(a) Adversary and discriminator losses

(b) Cycle losses

The left panel shows the average adversary and discriminator training losses (binary-crossentropy loss, as defined in (3.7)) between the two networks that are present in CycleGAN. As is typical for GANs, the losses oscillate—but at a relatively low level. The model training was stopped once the number of epochs surpassed 800 and all discriminator and adversary losses were below 0.5. The right panel shows how the cycle training losses (mean absolute error)—again averaged between the two networks—decrease during training.

The average cycle, adversarial, and discriminator training losses during 800 epochs of batch training are plotted in Figure 3.10. Further epochs were added in order to stop the model at a point where the discriminator and adversarial losses of both networks were lower than 0.5, in order to ensure that the generator and discriminator were balanced. These results stem from training a network that translates Indian English speakers' MFCCs to those of American English speakers. It is normal for losses to go up or down almost randomly when training GANs. This is due to the fact that we are training two competing networks and sometimes one or other of the networks takes over.

Figure 3.10 shows how the average cyclic training loss (measured using the mean absolute error) clearly decreased during training. A low cycle loss is important in order to ensure that the speech content does not change too much when translating to an American English accent. Also, since we put five times more weight on the cycle losses than on the adversary ones, we expected the average cycle training loss to decrease over training in the first place. The adversary and discriminator training losses (measured in terms of the binary-crossentropy loss, as defined in (3.7)) oscillate, but on a relatively low level.

The average cycle test error (out-of-sample) lies, at 8.27, in the range of the training errors during the final epochs. The average discriminator and adversary test errors (2.56 and 5.73 respectively) are, however, higher than the respective training errors when we stopped training (at 0.5). This is not surprising though if we look at the oscillating behavior of the discriminator and adversary training errors in Figure 3.10.

After translating all MFCCs of Indian English speakers—using the generator of the trained network—we feed them back to Kaldi and decode the modified MFCCs. At this stage, the decoding WER of modified Indian English speakers' MFCCs did not yet fall *below* that of non-modified MFCCs (see Figure 3.11). The WERs are, however, on a similar *level*. In addition, decoding utterances that have not been used for training the CycleGAN (out-of-sample utterances) results in a WER for Indian English speakers of 0.84 (before these utterances' MFCCs were modified, their average decoding WER was 0.82), indicating that the wrapper is generalizable.

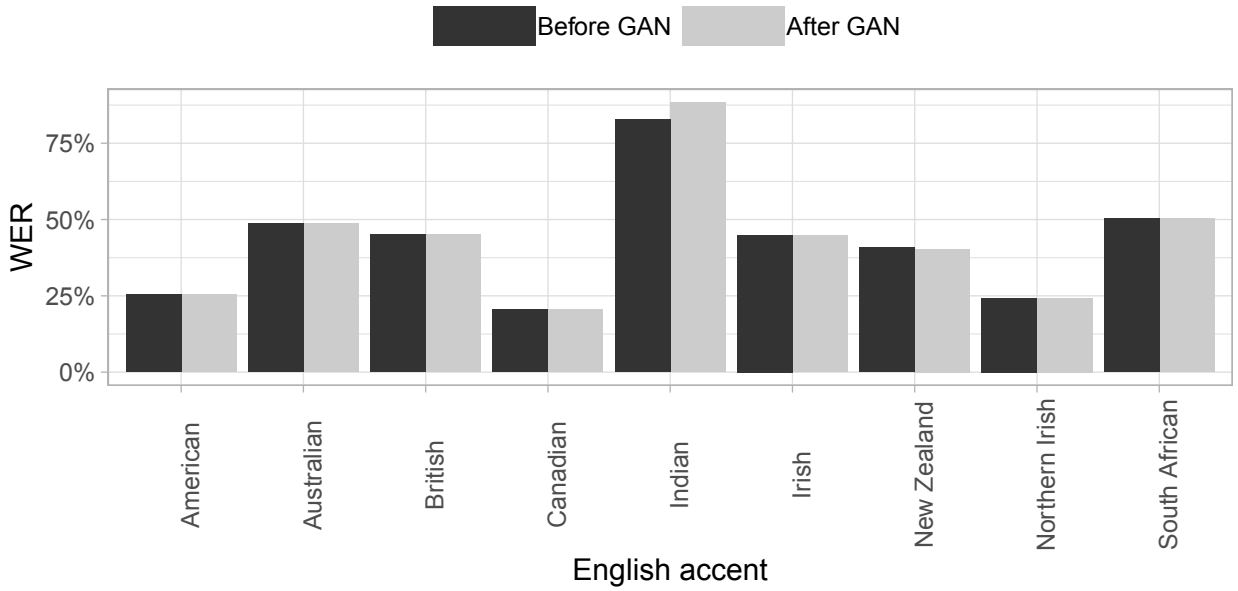
An issue at this stage is that the results are not very stable yet: training two identical networks with randomized training data can lead to very different models and, hence, decoding WERs.

3.6 Conclusion

The aim of this paper was to examine the fairness of automatic speech recognition (ASR) models. The first step was to show what concepts of fairness exist and how their measures can be adapted for ASR. From our analyses we conclude that ASR models that are trained on mel-frequency cepstral coefficients (MFCCs) suffer—by the nature of how MFCCs are set up—from disparate treatment: we have shown that the MFCC-distance of speakers with the same accent is clearly smaller than that of speakers with different accents. This implies that subjects' sensitive attributes are not ignored when ASR models are trained.

In addition, we analyzed how training corpus that are imbalanced with respect to the amount of speech per accent affect fairness. We have shown that systems perform worse in decoding

Figure 3.11: Decoding performance of VoxForge speech before and after GAN



The plot shows decoding performances per accent of an ASR model trained on LibriSpeech data. The dark gray bars indicate the WERs of decoding MFCCs the way they were outputted by Kaldi. The light gray bars show the result of passing Indian English speakers' MFCCs through a wrapper—our trained CycleGAN generator, whose goal it is to make Indian English speakers' MFCCs look like those of American English speakers. At this stage, the decoding error of the modified Indian English speakers' MFCCs did not yet fall *below* that of decoding non-modified MFCCs; the WERs are, however, comparable. In addition, we show that the wrapper is generalizable because it performs similarly well if we only look at utterances that have not been used for training the CycleGAN (out-of-sample).

the speech of speakers whose accents are underrepresented in the training corpus. Once we increased the amount of speech in underrepresented accents in the training set, the decoding WER for the respective speakers decreased. This implies that ASR models that are trained on an imbalanced speech corpus not only suffer from disparate treatment but also from disparate impact.

In order to combat these two problems of ASR models trained on MFCCs, and therefore make such models fairer, we suggest including a wrapper in the model, right before the MFCCs are decoded with the trained model. The goal of that wrapper is to make the MFCCs of speakers whose accent is underrepresented in the training corpus (in our case, Indian English) resemble those of speakers whose accent has predominantly been used for training the ASR

model (in our case, American English). To do so, we adapt the technique of generative adversarial networks (GANs). In our example, the generator focuses on producing MFCCs that are indistinguishable from those of American English speakers, while the discriminator tries to identify if MFCCs came from the generative model—hence, were generated—or from the real data. In order to ensure that the speech content of Indian English speakers' MFCCs is maintained when modified to American English-resembling MFCCs, our network not only needs to minimize generator and adversarial losses—as is the case in regular GANs—but also so-called cycle losses, which are the integral part of CycleGANs.

Modifying the MFCCs of Indian English speakers by running them through the generator of the trained CycleGAN and feeding them back to the ASR software Kaldi for decoding should result in lower word error rates (WERs) for the speech of Indian English speakers. Thus far, the WERs of decoding the modified MFCCs are, however, still slightly higher. From what we have experienced and read from other researchers about training GANs, we learned that this is a difficult task. We are convinced, though, that with continuous model tuning, at some point we will obtain the desired results.

Bibliography

Adomavicius, G., and Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge & Data Engineering*, *17*(6), 734–749.

Aggarwal, C. C. (2016). *Recommender Systems: The Textbook*. Springer International Publishing.

Amini, A., Soleimany, A. P., Schwarting, W., Bhatia, S. N., and Rus, D. (2019). Uncovering and Mitigating Algorithmic Bias through Learned Latent Structure. In *Proceedings of the 2019 aaai/acm conference on ai, ethics, and society* (pp. 289–295).

Anne, K. R., Kuchibhotla, S., and Vankayalapati, H. D. (2015). *Acoustic Modeling for Emotion Recognition*. Springer.

Baltrunas, L., and Amatriain, X. (2009). Towards Time-Dependant Recommendation based on Implicit Feedback. In *Workshop on Context-Aware Recommender Systems (CARS-09)*.

Barocas, S., and Selbst, A. D. (2016). Big Data's Disparate Impact. *California Law Review*, *104*(3), 671–732.

Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., ... Zhang, Y. (2019). AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. *IBM Journal of Research and Development*, *63*(4/5), 4:1–4:15.

Bridle, J. S., and Brown, M. D. (1974). An Experimental Automatic Word Recognition System. *JSRU Report*, *1003*(5), 33.

Buhalis, D. (2000). Tourism and Information Technologies: Past, Present and Future. *Tourism Recreation Research*, *25*(1), 41–58.

Bulten, W. (2017). *Getting started with generative adversarial networks (GAN).* `https://www.wouterbulten.nl/blog/tech/getting-started-with-generative-adversarial-networks/`. (Accessed on 13 January 2020)

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling*

and User-Adapted Interaction, *12*(4), 331–370.

Buttle, F. (2004). *Customer Relationship Management*. Elsevier Butterworth-Heinemann.

Celis, L. E., Deshpande, A., Kathuria, T., and Vishnoi, N. K. (2016). How to be fair and diverse? *Computing Research Repository (CoRR)*.

Centraal Bureau voor de Statistiek. (2018). *CBS Open data StatLine.* `https://opendata.cbs.nl/`. (Accessed on January 16, 2018)

Chen, Y.-N., Celikyilmaz, A., and Hakkani-Tür, D. (2017). Deep Learning for Dialogue Systems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics—Tutorial Abstracts* (pp. 8–14). Association for Computational Linguistics.

Civil Rights Act. (1964). *Civil Rights Act of 1964, Title VII, Equal Employment Opportunities*.

Cleverdon, C., and Keen, M. (1966). *Factors Determining the Performance of Indexing Systems*. Aslib Cranfield Research Project.

Crawford, K. (2017). *The Trouble with Bias.* Keynote from 5 December 2017. Neural Information Processing Systems (NIPS).

D-rt Groep B.V. (D-reizen & VakantieXperts). (2018). *LinkedIn.* `https://www.linkedin.com/company/d-rt-groep-b-v-d-reizen-&-vakantiexperts-/`. (Accessed on July 13, 2018)

Day, G. S. (2000). Managing Market Relationships. *Journal of the Academy of Marketing Science*, *28*(1), 24–30.

Deshpande, M., and Karypis, G. (2004). Item-Based Top-N Recommendation Algorithms. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 143–177.

Du, M., Yang, F., Zou, N., and Hu, X. (2019). Fairness in Deep Learning: A Computational Perspective. *arXiv preprint arXiv:1908.08843*.

Dubitzky, W., Granzow, M., and Berrar, D. P. (2007). *Fundamentals of Data Mining in Genomics and Proteomics*. Springer Science & Business Media.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. (2012). Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (pp. 214–226). ACM.

Goodfellow, I. (2016). *Generative Adversarial Networks (GANs).* `https://media.nips.cc/Conferences/2016/Slides/6202-Slides.pdf`. NIPS 2016 Tutorial.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 2672–2680).

Gruhn, R. E., Minker, W., and Nakamura, S. (2011). *Statistical Pronunciation Modeling for Non-Native Speech Processing.* Springer Science & Business Media.

Gurbanov, T., and Ricci, F. (2017). Action Prediction Models for Recommender Systems Based on Collaborative Filtering and Sequence MiningHybridization. In *Proceedings of the Symposium on Applied Computing* (pp. 1655–1661). ACM.

Hardt, M., Price, E., and Srebro, N. (2016). Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 3315–3323).

Horber, D. (2015). *Wohnungssuche auf Onlineplattformen: Diskrepanz zwischen der in den Sucheinstellungen angegebenen Preisobergrenze und der effektiven Zahlungsbereitschaft?* (Master thesis). University of Zurich, Switzerland.

Howard, A., and Borenstein, J. (2018). The Ugly Truth About Ourselves and Our Robot Creations: The Problem of Bias and Social Inequity. *Science and Engineering Ethics*, *24*(5), 1521–1536.

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 263–272). IEEE.

Kamiran, F., and Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, *33*(1), 1–33.

Koenecke, A., Nam, A., Lake, E., Nudell, J., Quartey, M., Mengesha, Z., ... Goel, S.

(2020). Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, *117*(14), 7684–7689.

Köhler, V. (2017). *Item-item collaborative filtering with binary or unary data.* `https://medium.com/radon-dev/item-item-collaborative-filtering-with-binary-or-unary-data-e8f0b465b2c3`. (Accessed on January 16, 2018)

Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, *43*(1), 59–69.

Kohonen, T. (2001). *Self-Organizing Maps* (Vol. 3). Springer.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *Computer*, *42*(8), 30–37.

Lee, D. D., and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*(401), 788–791.

Lee, D. D., and Seung, H. S. (2001). Algorithms for Non-Negative Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 556–562).

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of Massive Datasets.* Cambridge University Press.

Linden, G., Smith, B., and York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*(1), 76–80.

Little, R. J. A. (1988). A Test of Missing Completely at Random for Multivariate Data with Missing Values. *Journal of the American Statistical Association*, *83*(404), 1198–1202.

Little, R. J. A., and Rubin, D. B. (2014). *Statistical Analysis with Missing Data.* John Wiley & Sons.

Lloyd, S. P. (1982). Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2). Retrieved from `https://cs.nyu.edu/~roweis/csc2515-2006/readings/lloyd57.pdf`

Martel, A. (2018, February 13). Wissen, wer wo wohnen will. *Neue Zürcher Zeitung*. Retrieved from `https://www.realmatch360.com/wp-content/uploads/2018/02/Neue`

_Zrcher_Zeitung_2018-02-13.pdf

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2019). A Survey on Bias and Fairness in Machine Learning. *arXiv preprint arXiv:1908.09635*.

Mermelstein, P. (1976). Distance Measures for Speech Recognition—Psychological and Instrumental. *Pattern Recognition and Artificial Intelligence*, *116*, 374–388.

Messe Berlin GmbH. (2015). *ITB World Travel Trends Report 2015 / 2016.*

Miao, Y., Zhang, H., and Metze, F. (2015). Speaker Adaptive Training of Deep Neural Network Acoustic Models Using I-Vectors. *IEEE/ACM Transactions on Audio, Speech, and Language Processingyes*, *23*(11), 1938–1949.

Mnih, A., and Salakhutdinov, R. R. (2008). Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 1257–1264).

Molinaro, A. M., Simon, R., and Pfeiffer, R. M. (2005). Prediction Error Estimation: A Comparison of Resampling Methods. *Bioinformatics*, *21*(15), 3301–3307.

Müller, M. (2007). Dynamic Time Warping. In *Information Retrieval for Music and Motion* (pp. 69–84). Springer.

Nair, P. (2018). *The dummys guide to MFCC.* https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd. Medium. (Accessed on 13 January 2020)

Neue Zürcher Zeitung/Wüest & Partner. (2016). *Immo-Barometer.*

Newell, S. J., Belonax, J. J., McCardle, M. W., and Plank, R. E. (2011). The Effect of Personal Relationship and Consultative Task Behaviors on Buyer Perceptions of Salesperson Trust, Expertise, and Loyalty. *Journal of Marketing Theory and Practice*, *19*(3), 307–316.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: An ASR Corpus Based on Public Domain Audio Books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5206–5210). IEEE.

Pazzani, M. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, *13*, 393–408.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... Vesely, K. (2011). The Kaldi Speech Recognition Toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding.* IEEE Signal Processing Society.

Qin, Y., Mitra, N., and Wonka, P. (2018). How does Lipschitz Regularization Influence GAN Training? *arXiv preprint arXiv:1811.09567*.

Reichheld, F. F. (1993). Loyalty-Based Management. *Harvard Business Review*, *71*(2), 64–73.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 452–461). AUAI Press.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2011). *Recommender Systems Handbook*. Springer.

Rubin, D. B. (1978). Multiple Imputations in Sample Surveys - A Phenomenological Bayesian Approach to Nonresponse. In *Proceedings of the Survey Research Methods Section of the American Statistical Association* (Vol. 1, pp. 20–34). American Statistical Association.

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley.

Rubin, D. B. (2004). The Design of a General and Flexible System for Handling Nonresponse in Sample Surveys. *The American Statistician*, *58*(4), 298–302.

Sakoe, H., and Chiba, S. (1978). Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal processing*, *26*(1), 43–49.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 2234–2242).

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (pp. 285–295). ACM.

Sattigeri, P., Hoffman, S. C., Chenthamarakshan, V., and Varshney, K. R. (2019). Fairness GAN: Generating datasets with fairness properties using a generative adversarial network. *IBM Journal of Research and Development*, *63*(4/5), 3–1.

Schafer, J. L. (1997). *Analysis of Incomplete Multivariate Data*. Chapman and Hall/CRC.

Senior, A. (2017). *Speech Recognition.* `https://github.com/oxford-cs-deepnlp-2017/lectures/blob/master/Lecture%209%20-%20Speech%20Recognition.pdf`. Lecture at Oxford University.

Soltau, H., Liao, H., and Sak, H. (2016). Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition. *arXiv preprint arXiv:1610.09975*.

Staehelin, K., and Heiniger, B. (2017, August 13). Bauherren drohen Pleiten, Mieter profitieren. *Blick.ch*. Retrieved from `https://www.blick.ch/news/wirtschaft/so-viele-leere-wohnungen-wie-nie-zuvor-bauherren-drohen-pleiten-mieter-profitieren-id7296660.html`

Staub, P., and Rütter, H. (2014). Die Volkswirtschaftliche Bedeutung der Immobilienwirtschaft der Schweiz. *pom+, HEV Schweiz, Zürich*.

Sumpter, D. (2018). *Outnumbered: From Facebook and Google to Fake News and Filterbubbles - The Algorithms that Control our Lives*. Bloomsbury Sigma.

Tatman, R. (2017). Gender and Dialect Bias in YouTube's Automatic Captions. In *Proceedings of the First Workshop on Ethics in Natural Language Processing* (pp. 53–59). Association for Computational Linguistics.

Teich, D. A. (2018). *Machine Learning And Artificial Intelligence In Business: Year In Review, 2018.* `https://www.forbes.com/sites/davidteich/2018/12/26/machine-learning-and-artificial-intelligence-in-business-year-in-review-2018/#243e95882041`.

The State of Data and Analytics in Travel. (2017). *The State of Data and Analytics in Travel Report 2017.* EyeforTravel.

The U.S. Equal Employment Opportunity Commission (EEOC). (1979). *Adoption of Ques-*

tions and Answers To Clarify and Provide a Common Interpretation of the Uniform Guidelines on Employee Selection Procedures. https://www.eeoc.gov/policy/docs/qanda_clarify_procedures.html. Federal Register, 44 (43).

Veaux, C., Yamagishi, J., and MacDonald, K. (2017). *CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit.* https://doi.org/10.7488/ds/1994. University of Edinburgh. The Centre for Speech Technology Research (CSTR).

Vontobel, N. (2017, August 19). Da braut sich was zusammen: Jede 10. Mietwohnung ist ausgeschrieben. *Aargauer Zeitung*. Retrieved from https://www.aargauerzeitung.ch/schweiz/verpasste-chance-viele-mietwohnungen-werden-am-falschen-ort-gebaut-131625856

VoxForge. (2006–2020). http://www.voxforge.org/. (Accessed on 20 March 2019)

Wehrens, R., and Buydens, L. M. C. (2007). Self- and Super-Organizing Maps in R: The kohonen Package. *Journal of Statistical Software*, *21*(5), 1–19.

Weissenberg, A., and Langford, G. (2018). *Moving the global travel industry forward.* Deloitte. (Special article produced for the 2018 WTTC Global Summit)

World Travel & Tourism Council. (2018). *Travel & Tourism Economic Impact 2018.*

Wüest & Partner. (2015). *Immo-Monitoring 2015-2.*

Yang, K., Qinami, K., Fei-Fei, L., Deng, J., and Russakovsky, O. (2020). Towards Fairer Datasets: Filtering and Balancing the Distribution of the People Subtree in the ImageNet Hierarchy. In *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2020.* ACM.

Yi, X., Hong, L., Zhong, E., Liu, N. N., and Rajan, S. (2014). Beyond Clicks: Dwell Time for Personalization. In *Proceedings of the 8th ACM Conference on Recommender systems* (pp. 113–120). ACM.

Zafar, M. B., Valera, I., Rogriguez, M. G., and Gummadi, K. P. (2017). Fairness Constraints: Mechanisms for Fair Classification. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (Vol. 54, pp. 962–970).

Zhang, B. H., Lemoine, B., and Mitchell, M. (2018). Mitigating Unwanted Biases with Adversarial Learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (pp. 335–340). ACM.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of The IEEE International Conference on Computer Vision (ICCV)* (pp. 2223–2232).

Curriculum Vitae

Personal Details

First name, last name:	Vanessa Kummer
Date of birth:	19 October 1991
Citizenship:	Liechtenstein

Education

September 16–October 20:	PhD studies in Management and Economics, University of Zurich
April 14–September 16:	Master of Arts in Economics, University of Zurich
September 10–April 14:	Bachelor of Arts in Economics, University of Zurich

Professional Experience

Since October 15:	Teaching and research assistant at the Chair of Quantitative Business Administration, University of Zurich
March 15–August 15:	Internship in the pricing department at Basler Versicherungen AG
January 11–August 13:	Holiday work placements at the fiduciary office Jeeves AG
July 09–July 09:	Internship abroad at Oerlikon Balzers China